

# Banking Industry Architecture Network

## Service Landscape

## A How To Guide

Authors:	Guy Rackham, Dave Banko, Hans Tesselaar,
Version:	2.0.4
Last Change:	23 January 2013

## Copyright

© Copyright 2013 by BIAN Association. All rights reserved.

THIS DOCUMENT IS PROVIDED "AS IS," AND THE ASSOCIATION AND ITS MEMBERS, MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; OR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. NEITHER THE ASSOCIATION NOR ITS MEMBERS WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT UNLESS SUCH DAMAGES ARE CAUSED BY WILLFUL MISCONDUCT OR GROSS NEGLIGENCE. THE FOREGOING DISCLAIMER AND LIMITATION ON LIABILITY DO NOT APPLY TO, INVALIDATE, OR LIMIT REPRESENTATIONS AND WARRANTIES MADE BY THE MEMBERS TO THE ASSOCIATION AND OTHER MEMBERS IN CERTAIN WRITTEN POLICIES OF THE ASSOCIATION.

# Table of Contents

<b>Executive Summary .....</b>	<b>6</b>
<b>Part 1 - Introduction .....</b>	<b>7</b>
1.1 What is BIAN .....	7
1.2 BIAN's Motivation .....	7
1.3 Request for Feedback .....	8
<b>Part 2 - The BIAN Service Standards.....</b>	<b>9</b>
2.1 BIAN's Service-Oriented Architecture (SOA) Rationale .....	9
2.2 The BIAN Service Landscape - Overview .....	10
2.3 The BIAN Service Domain - Overview.....	11
2.3.1 The Completion Status of BIAN Service Domains .....	12
2.4 The BIAN Business Scenario - Overview .....	12
2.5 Applying BIAN Designs – Overview.....	13
<b>Part 3 - The BIAN SOA Design Framework.....</b>	<b>14</b>
3.1 Artefacts Used in the BIAN SOA Design Framework .....	14
3.1.1 The BIAN Service Landscape .....	15
3.1.1.1 Different Arrangements of the BIAN Service Landscape .....	16
3.1.1.2 Service Landscape Summary .....	18
3.1.1.3 Using the BIAN Service Landscape .....	19
3.1.2 The BIAN Business Scenario.....	19
3.1.2.1 Business Scenario Templates.....	20
3.1.2.2 Using the BIAN Business Scenario.....	21
3.1.3 The BIAN Service Domain .....	21
3.1.3.1 The Service Domain's Focus Object.....	22
3.1.3.2 Rightsizing the Service Domain .....	22
3.1.3.3 Translating High Level Semantic Designs into Software Specifications .....	22
3.1.3.4 Service Domain Templates .....	23
3.1.3.5 Message Level Mapping .....	28
3.1.3.6 Using the BIAN Service Domain Template .....	28
3.1.4 Data Repository Tool .....	28
3.1.5 The BIAN Metamodel and Business Vocabulary .....	29
3.2 Intended Uses of the BIAN SOA Design Framework .....	30
3.2.1 Applying BIAN Designs for Targeted Solutions.....	30
3.2.1.1 Select and Amend the Business Scenario .....	31
3.2.1.2 Define the Target State Requirements.....	31
3.2.1.3 Mapping Existing Systems to the Target State Business Scenario .....	32
3.2.1.4 Shortfall Impact Analysis.....	33
3.2.1.5 Solution Options.....	33

3.2.1.6	Example Business Uses .....	35
3.2.2	Applying BIAN Designs for Enterprise Planning .....	36
3.2.2.1	Building a Business Blueprint.....	36
3.2.2.2	Attributing a Business Blueprint .....	38
3.2.2.3	Examples of Blueprint Based Planning & Analysis .....	40
<b>Part 4 – BIAN Design Techniques.....</b>		<b>41</b>
4.1	Concepts behind the BIAN Standards .....	41
4.1.1	BIAN Uses a Service Oriented Architecture (SOA).....	41
4.1.2	Business Model versus Software Model – An Issue of Scale .....	43
4.1.3	Governing Service Domain Design Concepts.....	44
4.2	The BIAN Service Domain Design and “2 Cycle” Development Approach.....	45
4.2.1	Service Domain Operational Characteristics .....	45
4.2.2	Service Domain Design Techniques .....	46
4.2.2.1	Right sizing the Service Domain’s Focus Object .....	46
4.2.2.2	Assigning a Single Standard Functional Pattern.....	47
4.2.2.3	Confirming the Service Domain Role using Business Scenarios.....	48
4.2.3	The BIAN “2 Cycle” Development Approach.....	48
4.3	The BIAN Metamodel & Business Vocabulary .....	49
4.4	Positioning BIAN alongside Other Standards Organizations.....	50
<b>Appendix 1 - Glossary .....</b>		<b>52</b>
<b>Appendix 2 – Case Studies .....</b>		<b>54</b>
5.1	SunGard .....	54
5.1.1	Why BIAN Makes Strategic Sense for SunGard .....	54
5.1.2	The SunGard Approach to SOA and BIAN .....	54
5.1.2.1	The Benefits of BIAN for SunGard and its Customers .....	54
5.1.2.2	SODA: SunGard’s Interpretation of BIAN Guidelines .....	54
5.1.3	SunGard’s Ambit Core Banking .....	57
5.1.3.1	The History of Core Banking .....	57
5.1.3.2	SunGard’s Ambit Core Banking solution.....	58
5.1.4	Working with Erste Bank .....	59
5.1.5	Conclusion and Next Steps.....	59
5.2	Scotiabank .....	60

## Table of Figures

Figure 1 - The BIAN Service Landscape.....	10
Figure 2 - Service Domain Mechanics .....	11
Figure 3 - A Simple Business Scenario.....	13
Figure 4 - BIAN Service Landscape .....	16
Figure 5 – Function Oriented Service Landscape .....	17
Figure 6 - Organization Oriented Service Landscape .....	17
Figure 7 - The Service Landscape Excel Spreadsheet .....	18
Figure 8 - A Simple Business Scenario.....	20
Figure 9 - The BIAN Business Scenario Template .....	21
Figure 10 - The Service Domain Template Make-Up.....	23
Figure 11 - The Responsibility Item Region Of The SD Template .....	25
Figure 12 - The Service Operation Region Of The SD Template .....	26
Figure 13 - A Complete Service Domain Template.....	27
Figure 14 – Business Scenario View in Data Repository Tool.....	29
Figure 15 - Three Mappings: Elements, One-2-One, One-2 - Many .....	34
Figure 16 - An Enterprise Divisional Business Area Model.....	37
Figure 17 - An Enterprise Model With Service Domains Added .....	38
Figure 18 - Business Blueprint With Different Attributions .....	40
Figure 19 - Service Domain Decomposition.....	42
Figure 20 - Process Decomposition .....	43
Figure 21 – Asset Decomposition Model .....	47
Figure 22 - Functional Pattern Hierarchy .....	47
Figure 23 - Functional Pattern Descriptions .....	48
Figure 24 - 2 Cycle Development Approach .....	49
Figure 25 - The Service Domain Template With Yellow Fields.....	50
Figure 26 - BIAN in a Matrix with Other Standards Groups .....	51
Figure 27 - SunGard's Adoption of SOA and BIAN's Service Landscape .....	56
Figure 28 - SunGard's Top Down Approach to Service Identification .....	57
Figure 29 - SunGard's Bottom Up Approach to Service Identification .....	57
Figure 30 – Scotiabank's Repository Based on the BIAN Service Landscape .....	60

## Executive Summary

---

### ***BIAN's Mission***

The Banking Industry Architecture Network (BIAN) is a global not-for-profit association of banks and service providers formed with the intention of establishing banking industry standards to improve systems interoperability. BIAN's focus is on defining standard service operations typically found within an organization described as application-to-application (A2A). This How to Guide describes the BIAN standards, provides guidelines for their use and describes the main underlying design concepts.

### ***BIAN's Service Oriented Architecture (SOA) Design Framework***

BIAN's standards are captured within the context of a BIAN SOA Design Framework. Predominant business modelling approaches use a process representation of business activity. A process defines a business event in terms of a sequence of linked actions. A limitation of this representation is its ability to handle the typical variations in sequencing and rules/policies found between deployments. As a result, process models are best for commodity type activities.

To avoid the limitations of process models BIAN uses a different representation of business activity. The BIAN SOA model represents business as a collection of discrete non-overlapping elemental business capabilities and models business activity as patterns of service operation based collaborations between these capabilities. Unlike process steps, the elemental business capabilities, when correctly defined, are similar in terms of their business purpose/role in all deployments.

The BIAN SOA Design Framework is made up of the following key components:

- *BIAN Service Landscape* – a structured reference framework containing all identified elemental business capabilities
- *BIAN Service Domain Specifications* – the elemental business capabilities are termed BIAN Service Domains, their specifications outline their internal functions and the service operations that they offer and consume in operation
- *BIAN Business Scenarios* – present archetypical examples of Service Domains collaborating with one another through service operations in the execution of business

The BIAN interoperability standards are the generic elemental business capabilities (Service Domains) and their associated service operations. The service operations are specified in semantic terms that can be consistently interpreted in different technical implementations. BIAN interoperability standards are applied by aligning application functionality to the Service Domain boundaries. The Service Domain service operations then provide semantic definitions of the application interfaces.

### ***Reader's Guide***

This guide is divided into four Parts to target different audiences as described below:

**Part 1 - Introduction** - describes BIAN's intentions as a standards organization and sets out the overall layout of the guide

**Part 2 - The BIAN Service Standards** - provides an overview of BIAN's Service Operation standards, including BIAN's interpretation of a service oriented architecture (SOA). This Section is intended for all audiences

**Part 3 - The BIAN SOA Design Framework** - describes the different BIAN design artefacts that make up the BIAN SOA Design Framework and describes two different approaches for applying the designs – one for narrowly focused/targeted solutions and two as an enterprise analysis and planning framework. This section is intended for business and technical specialists responsible for applying BIAN designs

**Part 4 – BIAN Design Techniques** - describes the concepts and design principles that BIAN has developed and explains the design approach used within the BIAN membership to develop and maintain its standard definitions. This section is intended for SOA specialists that wish to better understand the theory underlying the BIAN approach

## Part 1 - Introduction

---

This document describes the BIAN Banking Industry Standards and how a financial institution or solution provider can apply them. It describes the different BIAN design artefacts and their intended uses and proposes techniques that can be used to interpret the designs in different technical environments. It also describes the design principles and approaches that BIAN uses internally to define standard semantic Service Operations to explain the theory behind the designs.

### *Available Documentation*

BIAN maintains a collection of documents that can be accessed on its website "[www.bian.org](http://www.bian.org)". Published versions are available to anyone at no charge. Access to working drafts is restricted to BIAN members.

The main BIAN documents include:

- The BIAN Service Landscape (and supporting Service Domain definitions)
- The BIAN How To Guide (this document)
- The BIAN Metamodel and Supporting Definition
- Selected BIAN Business Scenario Definitions
- Selected BIAN Service Domain Definitions (these include the BIAN standard semantic Service Operations)

### 1.1 What is BIAN

The Banking Industry Architecture Network (BIAN) is a not-for-profit association of banks, software vendors and service providers with the shared aim of defining Service Operation standards for the banking industry to improve software application interoperability within and between banks. BIAN's current focus is on internal application-to-application (A2A) interoperability.

### 1.2 BIAN's Motivation

BIAN's membership believes strongly that standard semantic designs for Service Operations will provide enormous benefit to the industry. When compared to a proliferation of proprietary designs, a standard view will:

- Enable the more efficient and effective development and integration of software solutions, primarily through greater solution re-use
- Support the adoption of more flexible business sourcing models
- Enhance the creation and adoption of shared third party business services

The BIAN initiative is iterative and depends on the active contribution of the banking community to build consensus and encourage adoption. BIAN seeks to coordinate the development of draft standards with supporting justifications and presents these to the industry, seeking feedback to refine the materials.

BIAN will continue to work with its membership using the feedback from the banking community to refine the presented specifications and expand the coverage across the BIAN Service Domains of the BIAN Service Landscape.

## 1.3 Request for Feedback

BIAN recognizes that feedback both from BIAN members and the banking community at large is essential for BIAN designs and design techniques to become a robust standard for the industry. This document presents candidate techniques that can be used to apply the emerging BIAN standards and the design theory and principles underlying the draft specifications for expert review.

The following process is in place to receive and process your feedback:

1. BIAN members are encouraged to provide feedback by using the BIAN Wiki, to the Architectural Committee, Service Landscape Working Group or via their Workgroup representatives
2. Non-members are invited to post their suggestions by using our website [www.bian.org](http://www.bian.org) or by contacting the chair of the Architectural Committee, Hans Tesselaar at [hans.tesselaar@bian.org](mailto:hans.tesselaar@bian.org)



## Part 2 - The BIAN Service Standards

---

This part of the guide provides a general overview of the BIAN standards. It is intended for all audiences.

The goal of BIAN is to improve application interoperability within and to a lesser extent among financial institutions, primarily banks. BIAN standards include semantic definitions of Service Operations that are structured in a consistent and well-defined fashion and include narrative descriptions of service operation behaviors.

To more clearly define Service Operations, BIAN has developed an approach to define formally general business partitions that offer and consume the Service Operations – called BIAN Service Domains. BIAN is seeking to scope out and specify all the Service Domains needed to support a modern bank. Once identified, the BIAN Service Domains are positioned in a reference structure called the BIAN Service Landscape.

The BIAN Service Landscape and its constituent Service Domains represent a type of 'Service Oriented Architecture' (SOA). BIAN's particular approach to SOA is briefly explained here with descriptions of its key ingredients: the BIAN Service Landscape; the BIAN Business Scenario; and the all-important BIAN Service Domain including its associated Service Operations.

BIAN maintains a comprehensive UML-based Metamodel of all its designs and design concepts to ensure their integrity and to help enable tooling support. BIAN is also developing and maintaining a business vocabulary. These more technical aspects of the BIAN Standards are explained in later sections of this document and documented within their own associated BIAN guidelines.

### 2.1 BIAN's Service-Oriented Architecture (SOA) Rationale

When a business and its supporting systems are aligned to the service partitions defined in a SOA the following types of benefits are expected:

- *The reduction of business information inconsistencies and fragmentation:* the SOA partitions are each intended to fulfil a unique and discrete business role and thus act as the single source for the services they provide and the business information that they 'govern.' The SOA therefore provides insights into function and information governance and usage that can be used to reduce inconsistency and fragmentation
- *Performance can be optimized:* because each service partition fulfils a narrowly defined purpose, its performance can be internally optimized
- *Service partitions can be widely accessed increasing re-use:* through its offered Service Operations, a service partition can be accessed across the enterprise. Concentrating scarce and specialized resources within the appropriate service partition, results in better resource utilization and capability re-use
- *Increased operational flexibility:* as more organizations align to a common definition of service partitions, underlying solutions can be more readily re-deployed and flexible operational sourcing arrangements and the business models that leverage them, can better be supported across the industry.

The BIAN SOA standards are defined independent of any particular organizational structure and technical implementation approach. A specific business enterprise can select and assemble Service Domain designs from the BIAN Service Landscape as needed to support its own particular processes and organizational layout, as described in more detail later in this guide.

## 2.2 The BIAN Service Landscape - Overview

The BIAN Service Landscape is a reference framework that contains all identified BIAN Service Domains. Its purpose is to provide a mechanism for quickly identifying and selecting Service Domains. The landscape uses a hierarchical decomposition of general banking industry capabilities at three levels:

1. **Business Area** – a broad area as might be associated with a major division of a bank
2. **Business Domain** – a smaller coherent set of capabilities within a business area that might correspond to a business unit or specialist business function
3. **Service Domain** – smallest element corresponding to a specific business need or function

It is intended that the BIAN Service Landscape should contain all possible Service Domains. Any business activity can be represented by a suitable collection of one or more Service Domains working together in collaboration. There are ~270 identified BIAN Service Domains of which ~210 can be identified as “Core Banking” functionality while the rest can be considered as “Business Support Functions.”

It should be noted that the current version of the BIAN Service Landscape defines some Business Areas at two levels: the Operations and Execution Business Area is further subdivided into four constituent Business Areas. The possible nesting of Business Areas is not captured formally in the BIAN Metamodel and definition of design terms but has been retained in this one case for continuity with prior designs.

### BIAN Service Landscape Structure

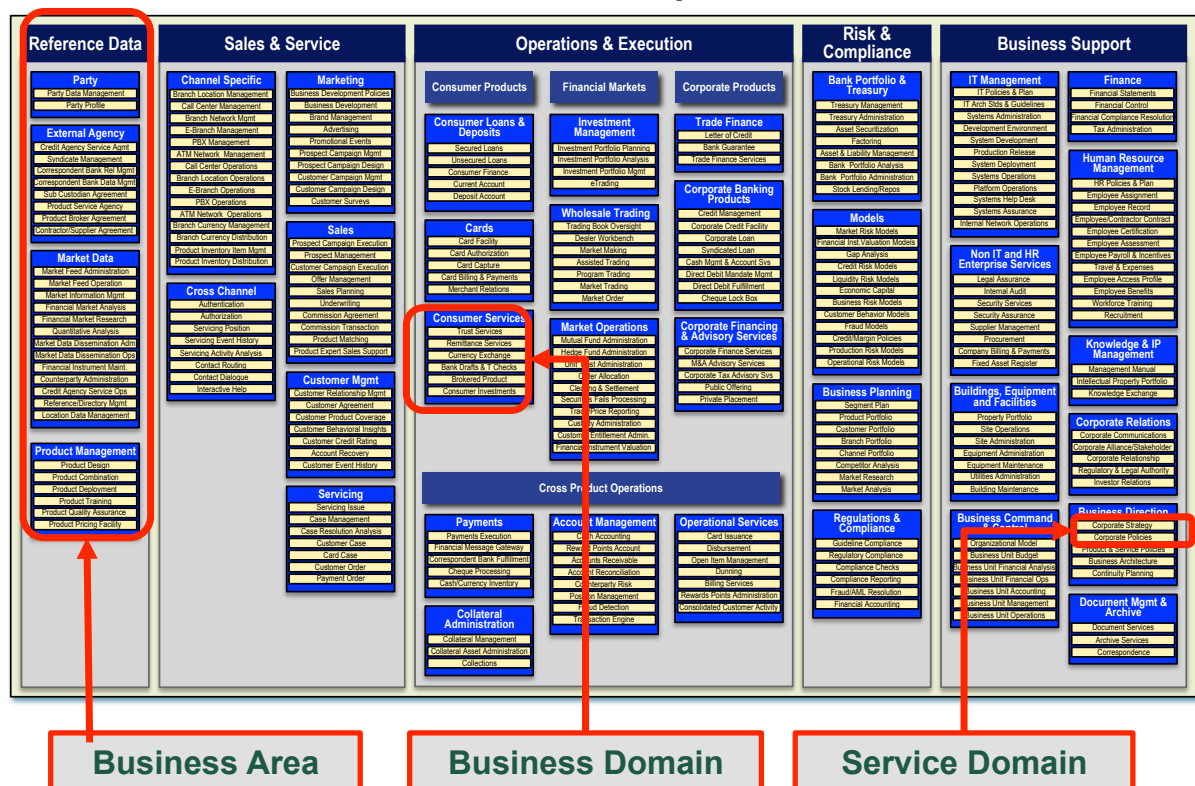


Figure 1 - The BIAN Service Landscape

## 2.3 The BIAN Service Domain - Overview

BIAN has developed a design rationale and supporting techniques to organize banking capabilities into non-overlapping partitions called 'Service Domains.' A BIAN Service Domain is designed to be 'elemental' meaning that it represents the smallest sensible service-enabled functional partition that supports a unique and discrete business role or purpose. All BIAN Service Domains taken together make up a 'peer set' that are arranged in a reference framework called the BIAN Service Landscape.

Some key Service Domain properties include:

- *A Unique Business Purpose* – a Service Domain has sole responsibility for fulfilling a specific and discrete business purpose
- *It is elemental* – it is not an assembly of other Service Domains. The full collection represents a 'peer' set of business capabilities
- *Collectively comprehensive* – all possible business activity can be modeled using Service Domains
- *Has a 'Focus Object'* – the focus object is a type of 'business object' that reflects the role or business purpose of the Service Domain. In simple (non-formal) terms, the focus object represents how the Service Domain exerts some type of control over some type of entity (e.g. handles the relationship contract for a customer)
- *Full Life-Cycle Support* – the Service Domain is responsible for all possible states of its focus object or for its full 'life-cycle' (e.g. inception, maintenance, operation, reporting and eventual closure/termination) – a focus object 'record' is used to trace the state of an occurrence of the Service Domain fulfilling its role from start to finish. Note the word 'record' is often dropped from the more precise term 'focus object record' in this text for brevity.
- *Single or Multiple Instances* – depending on the role, a Service Domain may need to handle a single active instance or multiple active instances of its focus object (for example a single business unit plan, or multiple customer accounts)
- *Short or Long Life-Span* – the life-span of a focus object instance can be short such as a customer interaction, or long such as a product design
- *Service Based* – all interactions with the Service Domain are realized through Service Operations and all possible business activity can be modeled as a pattern of service interactions between a suitable selection of Service Domains

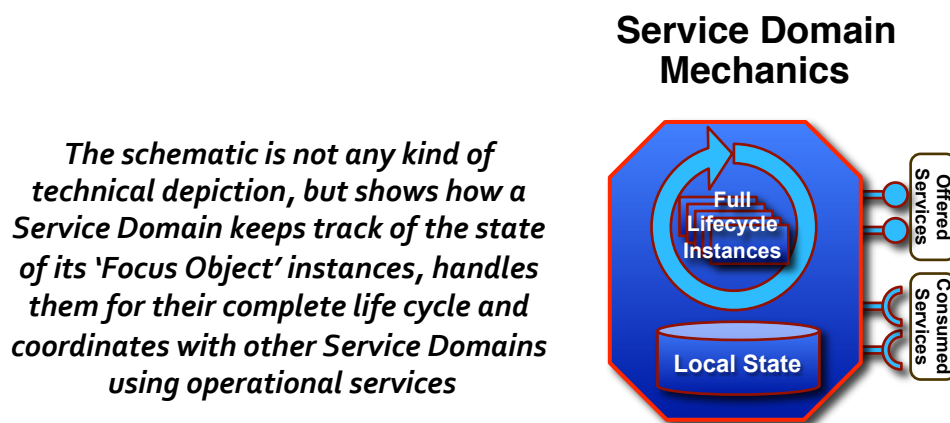


Figure 2 - Service Domain Mechanics

## 2.3.1 The Completion Status of BIAN Service Domains

BIAN describes the degree of completeness for Service Domains in the Landscape at three levels:

**Candidate** – General banking knowledge and/or analysis has indicated the need for a Service Domain. The Service Domain has been specified to a minimum level as captured in the supporting Service Landscape Excel spreadsheet with key details including a descriptive name, an outline of its business purpose/role, its focus object and a narrative example of business activity that involves it. These minimum design details are intended to confirm that the candidate Service Domain represents a discrete business capability that can operate as a service center.

**Defined** – The Service Domain has been included in at least one Business Scenario that has been worked through by a BIAN working group and the subsequent BIAN quality assurance procedures prior to publication. The analysis by the working group will have confirmed the business role performed by the Service Domain in the context of a specific and meaningful business event (as represented by the business scenario). It is important to note that the analysis associated with a single Business Scenario tests and refines the business purpose or role of the Service Domain but does not attempt to define the complete set of Service Operations that might be offered and consumed by the Service Domain. A 'defined' Service Domain therefore has a ratified business role/purpose and associated focus object, but is likely to have very few of its associated semantic Service Operations defined

**Complete** – Through multiple Business Scenario analyses or some other suitable mechanism, representative service operations covering all main activities and interactions involving the Service Domain have been captured and specified. This includes its offered and delegated service operations. It is important to note that BIAN seeks to define semantic standards for the primary activities of Service Domains that represent prevailing, common business practices. The business behaviors of many Service Domains will evolve as new differentiating services are pioneered. These leading practices will often become the prevailing practices of tomorrow. A 'complete' Service Domain therefore seeks to cover the established common practices but there can be additional Service Operations that represent unique or differentiating practices.

The current BIAN Service Landscape contains all currently identified Service Domains. As BIAN develops its standards, it is expected additional Service Domains will be identified and the scope and definition of existing Service Domains may need to be realigned. Version management will be used to maintain a trace of changes made to Service Domains between major releases.

## 2.4 The BIAN Business Scenario - Overview

Unlike the BIAN Service Landscape and BIAN Service Domain, the BIAN Business Scenario is not a formal design or canonical (i.e. the usual or standard representation), but instead is a simple depiction of how selected BIAN Service Domains might work together for some business event. It is used to help visualize the roles and Service Operations of Service Domains by example. BIAN uses Business Scenarios internally to help identify and specify Service Domains. Business Scenarios are also used in the application of BIAN designs as described in Part 3 of this guide.

## Simple BIAN Business Scenario

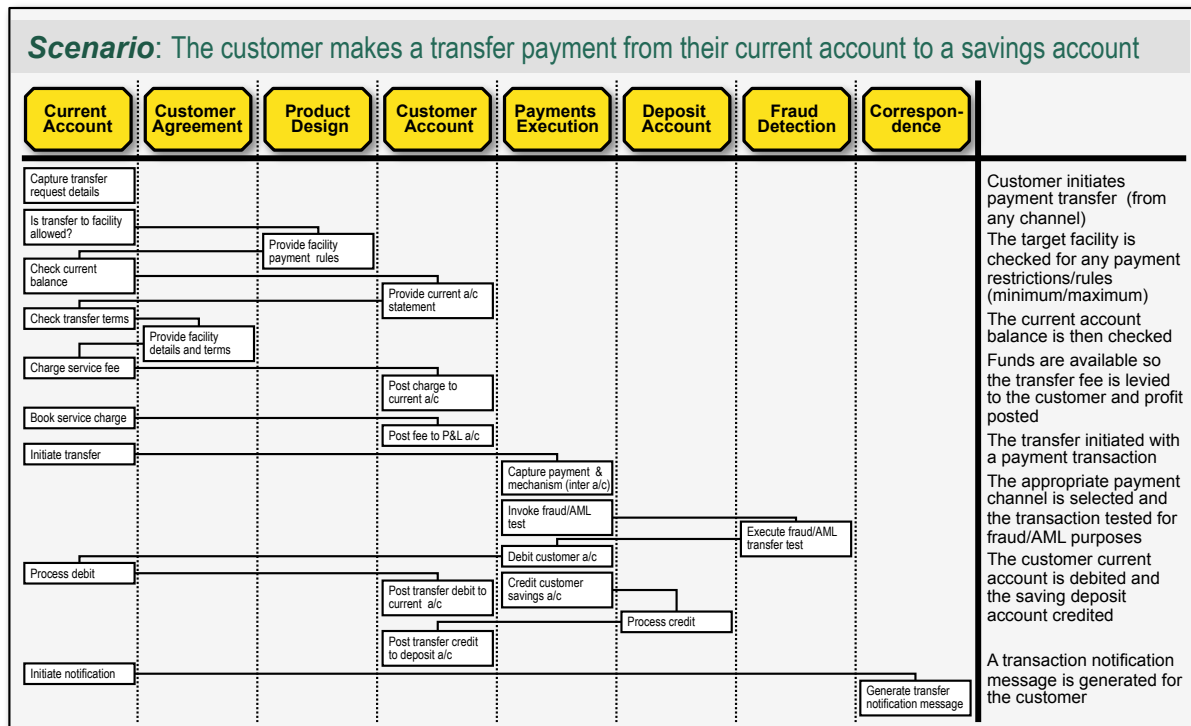


Figure 3 - A Simple Business Scenario

The flow of activity reads from top to bottom. In the diagram the role of a Service Domain and the actions it performs in the specific business scenario are captured in a single column. The lines connecting between columns show a Service Operation between the two corresponding Service Domains.

## 2.5 Applying BIAN Designs – Overview

The BIAN designs can be used at a financial institution to assess existing systems or as a high level blueprint for configuring packages or new application development. The BIAN Service Domains define best functional partitions that applications can be aligned to and the Service Operations correspond to the interfaces that the aligned applications need to support.

Two general types of use of the BIAN standards are expected. One to address targeted or narrowly focused needs where service oriented designs can enhance business performance, while another using the overall BIAN SOA as a framework for enterprise analysis and planning purposes.

The way BIAN designs are applied is described in more detail in Part 3.

## Part 3 - The BIAN SOA Design Framework

---

This part of the guide describes the different BIAN design artefacts including more detailed explanations of the BIAN Service Landscape, the BIAN Service Domain and the BIAN Business Scenario. It then describes how these design artefacts can be used for different types of improvement projects in different prevailing technical situations that can occur at financial institutions. The techniques described are intended for use by the business and IT architects of organizations using the BIAN SOA Design Framework as well as software vendors and service providers supporting the banking industry.

Section 3.1 of the guide describes the BIAN design artefacts making up the BIAN Design Framework as follows:

- 3.1.1 – The BIAN Service Landscape – is a reference structure for defining Service Domains
- 3.1.2 – The BIAN Business Scenario – provides an archetypical example of two or more Service Domains collaborating in response to a business event
- 3.1.3 – The BIAN Service Domain Template – is a structured specification of a BIAN Service Domain including its standard semantic Service Operations
- 3.1.5 – The BIAN Metamodel & Business Vocabulary – are critical supporting design assets used to maintain the integrity of the BIAN design standards

The descriptions in this Section are intended to provide an overview – more specific training guides are maintained with the individual design elements to support content development. The examples shown use Microsoft® Excel. With Version 2.0 BIAN is migrating the design content to a new UML compliant tool and data repository. The updated tooling will have a different presentation format but includes the same data fields. Updated templates for design capture and reporting will be detailed in a later release of the How to Guide.

Section 3.2 of the guide describes the intended uses of the BIAN SOA Design Framework. The uses divide between two main deployment approaches. The first, a 'targeted solution' approach, addresses a specific aspect of the business operation as might be represented as one or more business processes. It uses a narrow collection of Service Domain designs to represent the target state. The Service Domains and linking Service Operations provide a structure to assess existing capabilities and design service-based solutions. The 'targeted solution' approach is described in Section 3.2.1.

The second deployment approach described as 'enterprise planning' uses the full collection of applicable Service Domain partitions to create a 'bank on a page' view of the enterprise. This view can then be used as a framework to support a wide range of enterprise/departamental analysis and planning initiatives. The 'enterprise planning' approach is described in Section 3.2.2.

### 3.1 Artefacts Used in the BIAN SOA Design Framework

The BIAN SOA Design Framework is the name given to the collection of BIAN design artefacts. BIAN's initial mission is to develop standards for Service Operations for the banking industry focusing on internal application-to-application (A2A) interoperability. To do this BIAN has developed a design framework that is its own version of a Service-Oriented Architecture (SOA) that contains a comprehensive collection of all the business capabilities a bank might employ. Typically any one bank will only need a subset of this collection, for example because it supports only certain banking products.

Using the BIAN SOA Design Framework it is our goal that any business activity can be modelled as a pattern of collaboration involving a suitable selection of Service Domains. An example of such a pattern is represented using a BIAN Business Scenario. The BIAN Business Scenario clarifies the roles of BIAN Service Domains and their Service Operation exchanges by providing a contextual illustration. The BIAN Business Scenario is not a formal canonical design but merely an archetypical instance of one possible pattern of collaboration.

To fulfil its business role, a Service Domain interacts with other Service Domains through offered and consumed *Service Operations*. In practice a Service Domain will be involved in any number of



business scenarios but because it fulfils a discrete business purpose, the Service Operations it offers and consumes can be defined as a 'bounded' collection.

The BIAN Service Domain's specification contains the semantic definitions of all its offered Service Operations and provides references to the Service Operations it consumes from other Service Domains. The specification also describes the key functions it performs internally to clarify its business role/purpose as accessed through its Service Operations. The BIAN industry standard is the semantic definition of a Service Domain's Service Operations.

The BIAN SOA Design Framework can be summarized as follows:

- BIAN Design Principles, Techniques and supporting design capabilities (outlined in Part 4 of this guide)
- The BIAN Service Landscape – a reference framework that contains all identified BIAN Service Domains
- A collection of BIAN Business Scenarios – informal illustrations of BIAN Service Domain collaborations associated with handling business events
- The BIAN Service Domain Specification – a design template detailing the Service Domain's main functionality, the Service Operations it offers and the Service Operations it consumes from other Service Domains

Though BIAN's initial mission is to focus on defining standard Service Operations to improve systems interoperability for internal application to application ('A2A') interactions, the specifications themselves are system implementation agnostic. The BIAN Service Operation specifications include nothing specific to nor are they reliant on some feature of any particular technical architecture or solution approach.

A Service Operation specification defines the exchange in a structured format with narrative terms as might be described and understood by a business practitioner. It is intended to be unambiguous and comprehensive such that where appropriate it can be interpreted for supporting systems implementation without the need to specify additional business requirements. (Note that significant additional design effort will typically be needed to translate BIAN's high-level semantic Service Operations into the far more detailed code level message designs.)

The BIAN Service Landscape covers all banking activity including Service Domains typically having a high dependency on underlying systems support and others that have minimal need for highly integrated information systems. Given BIAN's priority on improving systems interoperability, the designs and associated techniques described relate primarily to those Service Domains with a high systems dependency.

### 3.1.1 The BIAN Service Landscape

The BIAN Service Landscape is a reference structure that categorizes and organizes BIAN Service Domains for ease of access. Different criteria can be used to classify and organize Service Domains that would result in different layouts of the standard set of BIAN Service Domains. BIAN uses a 'primary' Service Landscape view based on agreed categorizations that have been refined in use by the BIAN membership.

The BIAN Metamodel is a detailed and comprehensive (UML) model that defines all the BIAN design structures – it is fully documented elsewhere in its own guide (The BIAN Metamodel). The Metamodel has three elements that capture the design of the BIAN Service Landscape.

1. *Business Area* – is the highest-level classification. A business area groups together a broad set of business capabilities. For the BIAN Service Landscape they are defined to be aspects of business activity that have similar supporting application and information-specific needs.

Note – Within the Business Area Operations and Execution of the current Service Landscape, four sub Business Areas have been defined. This is a historical feature and is not reflected in the BIAN standard design techniques and metamodel.

2. **Business Domain** – at the next level, business domains define a coherent collection of capabilities within the broader business area. In the BIAN Service Landscape the business domains are associated with skills and knowledge recognizable in the banking business.
3. **Service Domain** – is the finest level of partitioning, each defining unique and discrete business capabilities. The Service Domains are the ‘elemental building blocks’ of a service landscape

The Service Domain relates to generic capabilities that do not vary in their scope, but the definitions of the Business Domain and Business Area are classifications that are specific to a particular Service Landscape layout. In the BIAN Service Landscape Version 2.0 Business Areas have been defined corresponding broadly to types of systems use. It has the following Business Areas working from left to right:

- **Reference Data** – contains business domains (and their contained Service Domains) that handle access to both internally and externally sourced information that is widely accessed by different parts of the business like Party data
- **Sales & Service** – brings together business domains that support the interactions with the bank’s customers through all channels for purposes of selling and servicing in-force products and services
- **Operations & Execution** – is a large area that combines all transaction processing oriented aspects of product and service fulfilment, including product specific activities, ‘vanilla’ capabilities that can be integrated within many products and shared supporting operational services. Due to its large size, this area is subdivided into ‘sub’ Business Areas corresponding to product types and shared support capabilities
- **Analytics & Risk** – consolidates the business domains that support and perform detailed analysis functions. These cover product and customer related analyses, business unit performance assessments and all dimensions of risk (e.g. credit, market, instrument, operational & compliance)
- **Business Support** – combines the wide range of general management and support activities common to most enterprises, including the executive, finance, staff, systems and facilities

### 3.1.1.1 Different Arrangements of the BIAN Service Landscape

Within these five broad Business Areas, approximately 40 Business Domains represent generally recognizable banking functional groups. The approximately 270 elemental Service Domains have then been ‘loaded’ into this two-tiered reference framework.

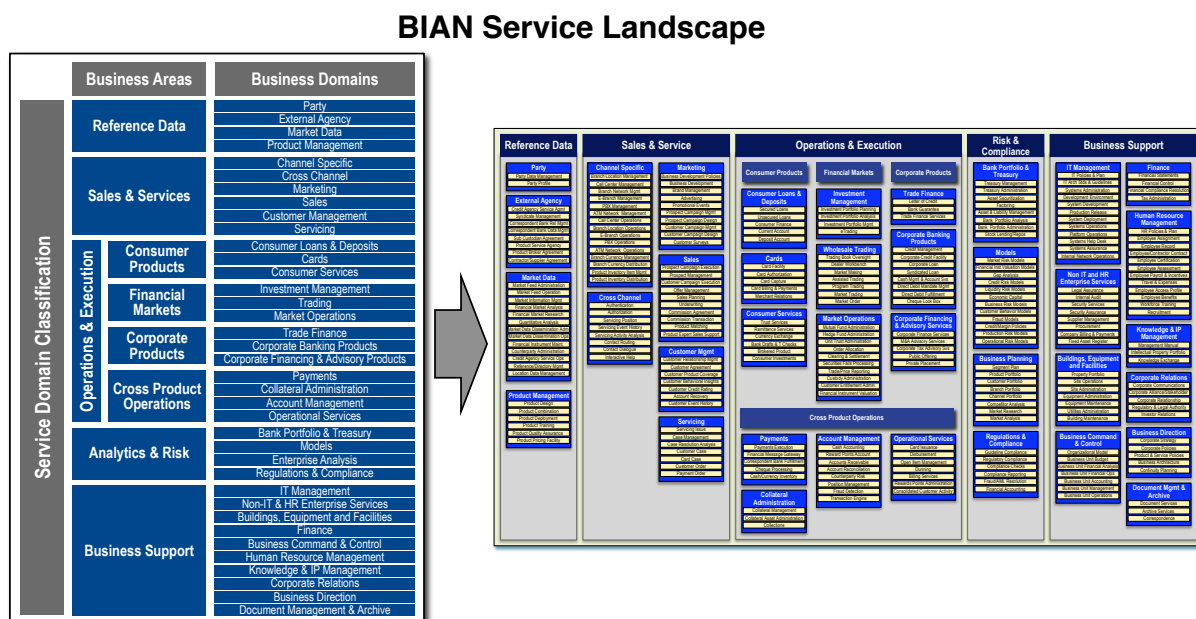


Figure 4 - BIAN Service Landscape



It is worth restating that the BIAN Service Domains are intended to represent the complete set of elemental business capabilities. By associating other characteristics to the definition of Business Areas and Business Domains different reference structures can be developed. Two simple examples of alternative layouts are shown next.

In the first alternative layout the Business Areas and Business Domains define a functional hierarchy, representing the types of activity performed. By grouping Service Domains this way supporting applications with similar functionality can be identified possibly helping with designing solutions for utility type re-use.

### Function Oriented Service Landscape

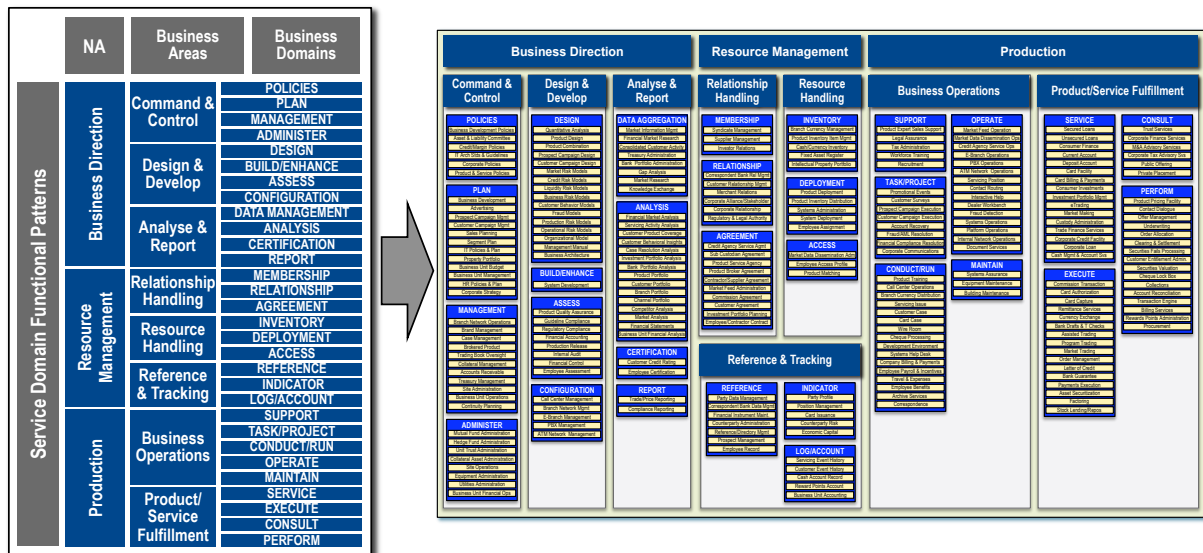


Figure 5 – Function Oriented Service Landscape

In the second alternative layout Business Areas and Business Domains reflect the operational partitions typically found in the business organization of a full service bank. An Organization Oriented Service Landscape can be helpful when considering organizational and management aspects of the Bank.

### Organization Oriented Service Landscape

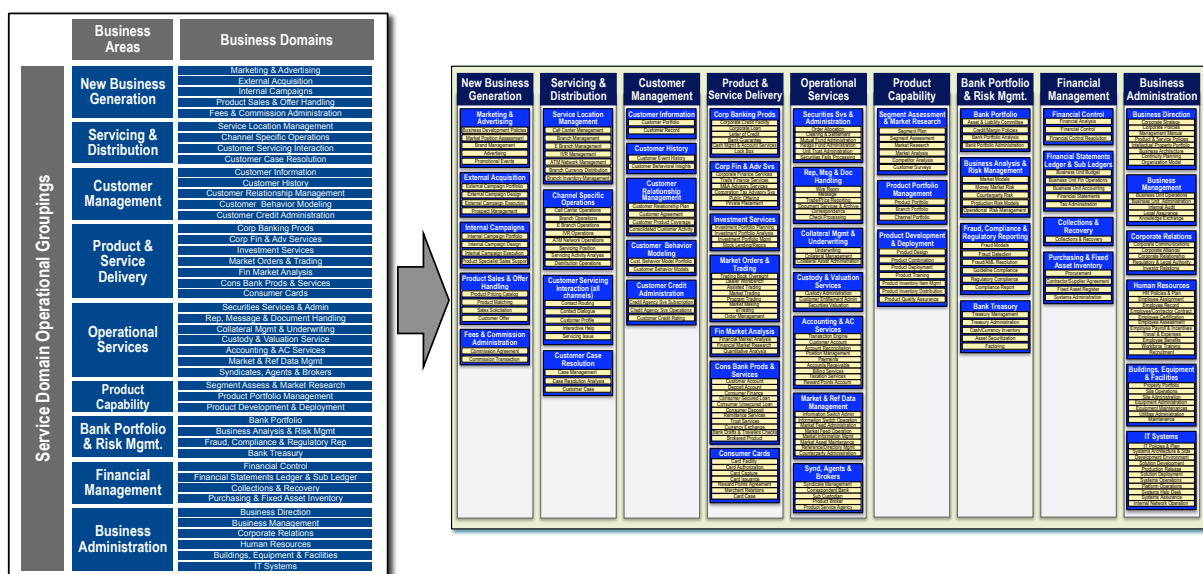


Figure 6 - Organization Oriented Service Landscape

### 3.1.1.2 Service Landscape Summary

There is an associated spreadsheet that contains basic descriptions for all Service Domains sorted by Business Area and Business Domain.

This overlaps with the content of the Service Domain Definitions, but provides a convenient summary of the role of all Service Domains in a single reference document. This content is currently maintained manually but in the future will be maintained in a UML compliant repository.

Refer to Part 4 of this guide for an explanation of technical terms used in these brief descriptions:

<i>Type</i>	specifies whether the row refers to a Business Area (BA), Business Domain (BD) or a Service Domain (SD)
<i>Name</i>	the descriptive name of the Service Domain
<i>Version 2.0 Business Role</i>	a brief description of the Service Domain's business role
<i>Focus Object</i>	a business object whose instances are managed by a single Service Domain
<i>Functional Pattern</i>	BIAN has observed repeating functional patterns for different Service Domains. There are currently 24 patterns.
<i>Comment</i>	general clarification of the role and any open considerations arising from BIAN's internal design discussions

## Service Landscape Excel Spreadsheet

A	B	C	D	E	F	G	H	I	J
BIAN Service Landscape V2.0									
Type	Service Domain Name	Business Role	Focus Object	Generic Object Class	Object Qualifier(s)	Functional Pattern	Reference Business Objects	Example of Use	General Comment
B	Reference Data								The area Reference Data contains all categories of managed business information, covering customer details, business partners, product details that is widely accessed across other activities
B	Party								This Domain in business area Reference Data combines both customer (consumer and corporate) and commercial type relationships.
S	Party Data Management	Maintain a comprehensive set of customer details, including demographics, administrative, KYC related, status and activity summaries	Customer Reference Details	Party>Customer		REFERENCE DETAILS	Party Relationship,	Customer details are obtained for selection in an internal sales campaign	This service domain maintains a comprehensive set of customer reference information. This is required for customers, but has significantly different content and functionality depending on customer type. BIAN currently combines the maintenance of customer reference data with all types of business relations with the single Party concept. The BIAN Party team need to determine whether a shared Party application means a single Service Domain for all types of Parties given that the profile of information managed varies greatly for different
S	Party Profile	Maintain a small amount of current customer status data to identify and influence any real-time operational interaction for sales, servicing and/or fulfillment (e.g. high value, eligible for campaign, account suspended,	Customer Indicator	Party>Customer		INDICATOR		The identity and active status of a customer is checked as they present themselves on-line to access their bank account	Party team need to ratify the operational nature of the Party profile as well as address the granularity issue noted for Party. The candidate service domain tracks key relationship status and detail information used to verify identity and support any type of real-time interaction with the customer/partner (such as account status, active case or customer campaign status). The profile data and role of the service domain will vary greatly between customer type parties and commercial alliances
B	External Agency								This Domain handles the range of arrangements with various external agencies

Figure 7 - The Service Landscape Excel Spreadsheet

More comprehensive specifications of the BIAN Service Domains are captured in the BIAN Service Domain template as described later in this part of the guide.

### **3.1.1.3 Using the BIAN Service Landscape**

The primary use of the Service Landscape is as a reference framework to organize the large collection (approximately 270) of Service Domains. By using different characteristics to organize the Service Domains (through the definition of Business Areas and Business Domains), many different similarities and associations can be revealed between the resulting groupings of Service Domains as the above examples show. In general terms the Service Landscape can be used as follows:

1. To find a specific Service Domain – the Business Area and Business Domain classifications provide a path to narrow in on the target Service Domain
2. To reveal similarities between Service Domains – the Business Area and Business Domain classifications can be defined to cluster related domains. The first example above does this based on similar functional behavior, the second on common organizational ownership
3. As a management planning framework - the Service Landscape presents a complete picture of all of the elements that make up the enterprise and can be used as a management framework to overlay many perspectives (see Section 3.2 of this guide for more detail).

### **3.1.2 The BIAN Business Scenario**

The BIAN Business Scenario is a visual representation of how some Service Domains might collaborate together in response to a business event. Its purpose is to clarify the different roles of Service Domains by example, and is not canonical or part of the BIAN standard. It defines an archetypical flow, the sequence as represented can change in practice, some interactions might be obsolete and others might be missing. In theory there is no practical limit to the number of business scenarios and possible variations that could be assembled using the standard BIAN Service Domains. BIAN uses combinations of Business Scenarios to clarify and explain the roles, boundaries and exchanges between Service Domains.

It's not BIAN's aim and intention to provide a complete set of Business Scenarios. Business Scenarios are an aid to populate and one way to reference the contents of the Service Landscape.

## Simple BIAN Business Scenario

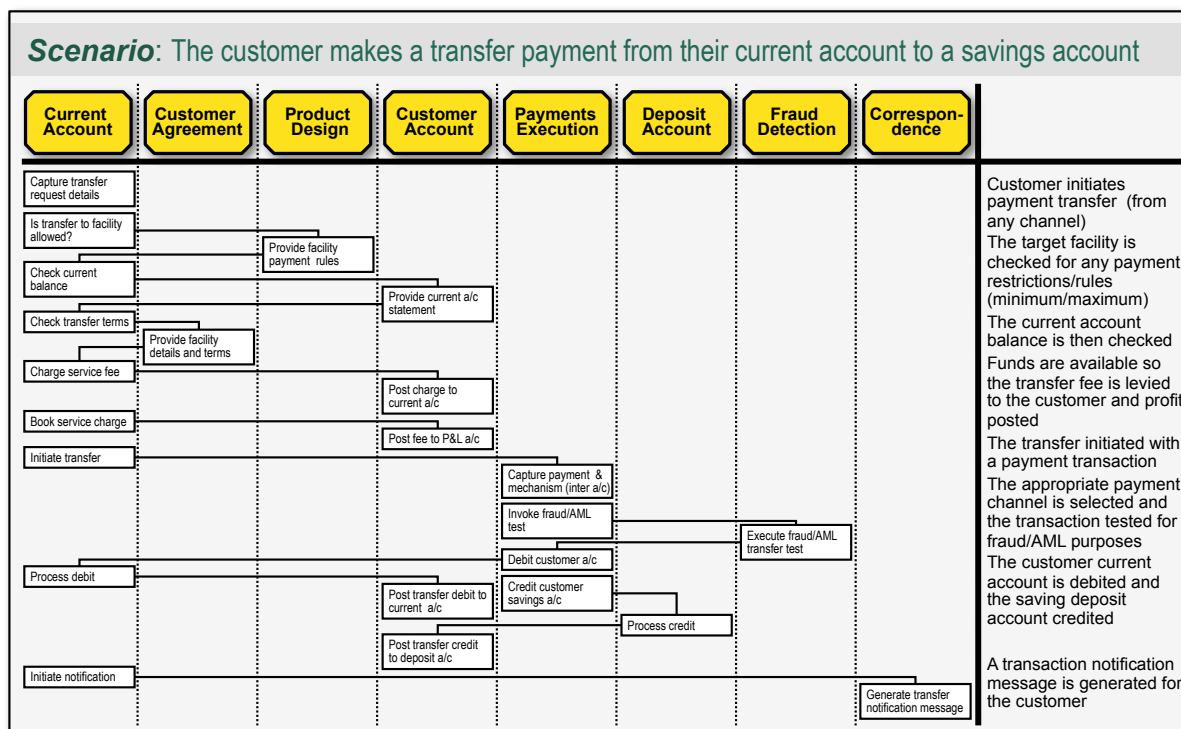


Figure 8 - A Simple Business Scenario

Each column in a Business Scenario diagram corresponds to the role played by one Service Domain. The implicit flow of activity is read from the top down. Boxes within each column contain text summarizing the internal functions performed and the connections between columns represent the Service Operation exchanges between Service Domains. In more detailed versions of the Business Scenario template the listed functions and Service Operations cross-reference the Service Domain specification (See the Service Domain Template).

The scope of a Business Scenario can be compared to that of a conventional high-level Business Process, but there is one key difference between a traditional Business Process representation and the Business Scenario. Both describe action steps and some implicit flow of control, but the Business Process does not formally divide functionality between discrete service based partitions (Service Domains) unlike the Business Scenario. Fortunately this very significant difference is largely transparent to the typical business practitioner who will recognize the actions and implicit sequence of events regardless of any particular grouping and layout.

The Business Scenario can be a good vehicle to introduce the BIAN service oriented designs. It is first used in a manner similar to a conventional Business Process, representing the flow of activity associated with a familiar business event in a form that can easily be recognized and understood. The same Business Scenario representation can then be used to explain the concept of Service Domains by exposing the discrete role each plays in handling the selected business event.

### 3.1.2.1 Business Scenario Templates

When using business scenarios to develop BIAN designs, a more formal structure is used to capture additional properties of the interactions between Service Domains. This template and brief descriptions of its content is shown here for reference:

## BIAN Business Scenario Template

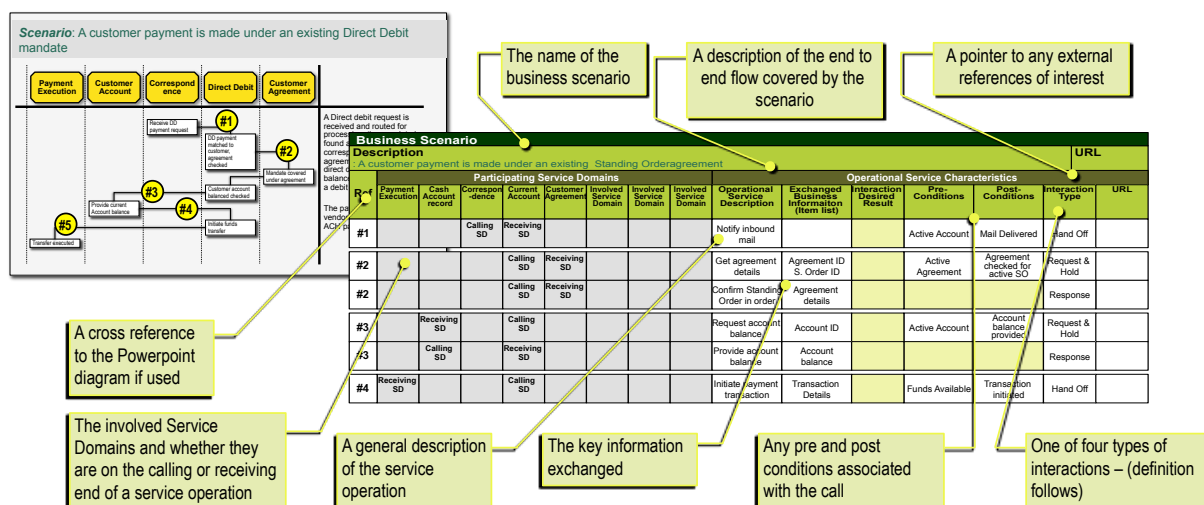


Figure 9 - The BIAN Business Scenario Template

One field of specific interest defines the type of interaction involved in a Service Operation interaction. Four types of exchange are currently defined:

- **Request & Hold (for Response)** – a simple two way exchange of information where the calling Service Domain waits for some response before continuing that thread of activity
- **Request & Monitor (for Response)** – a more complicated two way exchange, where the response is expected some significant time in the future. The caller must monitor for the response
- **Hand-Off** – a one way exchange, the caller initiates actions, but does not need to wait to see the result
- **Make Announcement (could be just one receiver)** – one or more Service Domains have subscribed to updates and the handling Service Domain generates suitable notifications when appropriate

The descriptions of the types of exchange help underscore the level at which the BIAN standards define Service Operations. They describe operational behaviors and dependencies but do not define the messaging protocols of any underlying systems that may be implemented to support aspects of the interaction. (See Section 4.4 Positioning BIAN alongside other Standardization Bodies)

### 3.1.2.2 Using the BIAN Business Scenario

The service domain partitioning that is at the heart of a service oriented architecture (SOA) represents a significant departure from the conventional process oriented designs that many are familiar with. Presenting Service Domains in the context of a familiar business operation using the Business Scenario format can make these service-based designs easier to understand. When several different business scenarios each involving a common Service Domain are presented this can also demonstrate how a service oriented architecture can be used to define highly re-usable and leveraged capabilities.

BIAN Business Scenarios can be matched to the internal processes at a bank and the Service Domains and their associated Service Operation boundaries used as an assessment framework. As noted the BIAN scenarios are indicative and in matching them to a specific location their sequencing and content may need to be changed to reflect the prevailing business rules and practices. As long as the discrete roles of the Service Domains remain consistent the associated Service Operation standards will apply.

### 3.1.3 The BIAN Service Domain

The BIAN Service Domain is the key building block for the BIAN standards. Each Service Domain 'offers' an associated standard set of Service Operations. The specification of the Service Domain and



its Service Operations is intended to be generic or 'canonical', meaning that its business role or purpose can be consistently interpreted in different banks. This characteristic is clearly critical for the definition of standards. Some other key considerations and properties of the Service Domain are described next to provide some additional explanation and context before the BIAN Service Domain template is detailed.

### 3.1.3.1 The Service Domain's Focus Object

As described briefly in Section 2.3 each Service Domain has an associated 'focus object'. A Service Domain's focus object represents its main business purpose or role. A role combines two elements: one is some type of influence or control exerted; the second is the business entity of object operated on. For example, applying a contractual agreement to a customer relationship. The focus object tries to succinctly capture the two elements. Here the control exerted is the agreement and the business entity is the customer leading to 'customer agreement' being the focus object.

The Service Domain is also defined to be responsible for exerting its control over the business entity for a full life-cycle. For a customer agreement, this corresponds to the initial set-up, all maintenance, any updates and providing on-going access to the agreement, also any associated analysis and reporting and the eventual termination of the agreement.

Depending on the role of the Service Domain the typical life cycle of its focus object can be short lived such as a contact with a customer (through a call center, on-line or at a branch) or long lived such as the specification of a product. It may be appropriate for a Service Domain to handle a single instance of its focus object, such as the budget for a business unit, or multiple concurrent instances of its focus object such as the in-force contractual agreements for customers. In the case where there are several active instances, the focus object is really a pattern that is applied to each instance independently.

Correctly isolating and defining the focus object is one of several design mechanisms BIAN uses to specify Service Domains that are generic. The current techniques are detailed in Part 4 of this guide. One other technique is outlined here as it helps provide further context for the description of the BIAN Service Domain template.

### 3.1.3.2 Rightsizing the Service Domain

A Service Domain needs to have its business scope 'right sized.' BIAN seeks to define the scope of the Service Domain to cover a single business need at the finest level of detail where it retains *unique business context*. Experience has shown that Service Domains scoped to this level of granularity are more easily designed to be generic.

Consider an example: the area of customer management clearly combines a broad range of business activities. Within customer management one of many activities that appears 'elemental' in nature is the handling of the customer agreement/contract as already mentioned. To test whether it is indeed a capability at the 'finest level of detail that retains unique business context' we can try to break it down further. This results in actions such as reviewing documents and maintaining customer details that are not uniquely associated with handling a customer agreement but are more utility in nature (i.e. the same activity could be performed elsewhere for a different purpose). Handling customer agreements is therefore according to this one technique a good candidate capability for a right-sized Service Domain.

### 3.1.3.3 Translating High Level Semantic Designs into Software Specifications

The Service Domains and their associated Service Operations are formally defined in semantic, structured terms at a high level in the Service Domain template. There will inevitably be site-specific variations and different technical implementation options as these high level designs are expanded to software level implementation specifications. As long as the resulting application boundaries align to the BIAN Service Domain boundaries and the application interfaces reflect the BIAN standard Service Operations the expected benefits from alignment can be realized (including optimized performance, improved interoperability, better resource leverage and greater solution reuse. The retention of the Service Domain boundaries in the derived software design is an important aspect of implementing a service oriented architecture that has no equivalent in conventional process based designs. In process based design the high level process steps are broken down to progressively finer levels of detail until

they match the necessary precision for coding. Through this decomposition, any distinction of the high level partitions between steps in the process typically gets 'dissolved away' at lower levels.

In contrast, service-oriented design retains the functional boundaries defined by the Service Domains. These boundaries typically correspond to major application or application module boundaries. The capabilities performed within the Service Domain represent the internal application functionality (that can be implemented using any appropriate design approach) and the Service Operation interactions between Service Domains correspond to major application interfaces. The conversion of the high level semantic designs to the software specification adds significant detail and specificity to the function and service designs but retains the original Service Domain partition.

### 3.1.3.4 Service Domain Templates

The BIAN Service Domain template has two main regions. The header region includes fields that provide general overview details and the main body of the template lists the Service Domain functionality and Service Operations offered and consumed organized into four general types.

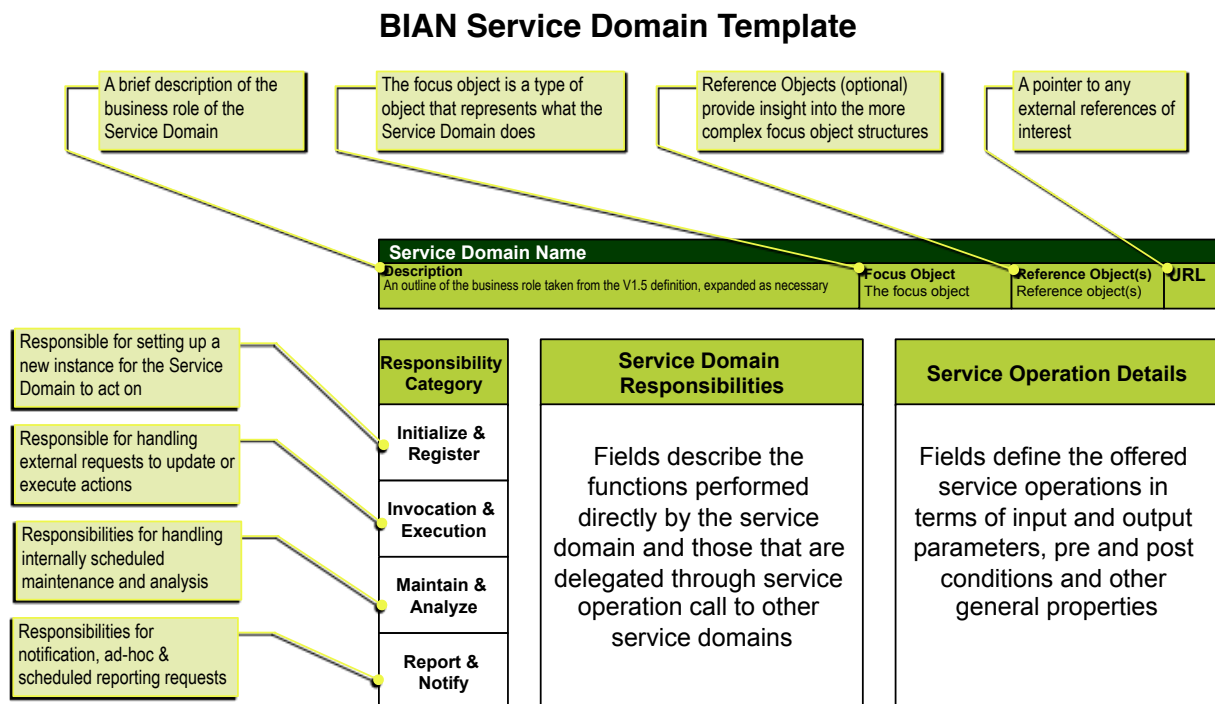


Figure 10 - The Service Domain Template Make-Up

The main fields and regions of the template are:

<i>Description</i>	contains one or two sentences to describe the business role of the Service Domain
<i>Focus Object</i>	is a control record that is used to keep track of one instance of the Service Domain fulfilling its business role from start to finish
<i>Reference Object</i>	Is an optional field to capture one or more conceptual business objects that clarify the structure of more complex focus objects
<i>URL</i>	is a field where pointers to any supporting documentation of interest is captured (typically in the public domain)
<i>Responsibility Category</i>	defines four general groupings for capturing the internal functionality and associated Service Operations as described in the diagram above

<i>Service Domain Responsibilities</i>	is a region where the internal functionality is listed in a format that aligns with offered and consumed Service Operations – explained in more detail below
<i>Service Operation Details</i>	is another region where the offered Service Operations are specified – explained in more detail below

### 3.1.3.4.1 **Reference Object**

The Reference Object field is used to capture one or more business objects that clarify or elaborate on the definition of the Service Domain's focus object. For example if the generic object is a complex entity it can be helpful to list additional business objects to clarify its structure. If the Service Domain provides or depends on critical business information, this too can be indicated using the Reference Object field.

The Reference Object list captured in the Service Domain template is a 'stop-gap' measure to capture important design details in anticipation of a more comprehensive design solution. Service Domain specifications are likely to eventually include some form of conceptual business object model to clarify the structure of the focus object and to help define the Service Domain's business information scope of governance and its key information dependencies.

When developed the Service Domain's conceptual business information specification will also provide a more comprehensive and formal definition of many of the input and output parameters that make up the semantic service operation specifications.

### 3.1.3.4.2 **Responsibility Categories**

BIAN is not attempting to define fully the general functionality of a Service Domain but instead concentrate on defining the Service Operations it offers and consumes as the basis for establishing interoperability standards. However to define Service Operations some description of the internal Service Domain functionality is necessary. BIAN defines Service Domain functionality in terms of the activities for which it is responsible and it needs to execute its business role. Some activities it performs itself using its own internal capabilities, for others, it might need to delegate some aspect of execution to other Service Domains.

As noted in the outline description of the template above, the grouping of responsibility items into four general categories is only to ease reference:

- **Initialize & Register** – activities associated with setting up, verifying and/or registering a new occurrence of the Service Domain's focus object
- **Invocation & Execution** – activities associated with the Service Domain performing tasks in response to an external request
- **Maintain & Analyze** – activities associated with internally scheduled analysis, maintenance, housekeeping and upkeep
- **Report & Notify** – activities associated with providing schedule and ad-hoc reporting or responding with notifications to subscribing Service Domains

### 3.1.3.4.3 **Service Domain Responsibilities**

The responsibility of the Service Domain is described as more fine grained 'responsibility items' where each item uses a single row of the template.

There are three reasons to define a responsibility item for a Service Domain:

1. The responsibility item handles a Service Operation
2. The responsibility item delegates activity to a different Service Domain
3. The responsibility item describes additional activity for clarity



One responsibility item (row) is defined for each offered Service Operation and each time internal processing delegates to another Service Domain, which can be triggered by the processing following an external Service Operation call or by internal scheduling. Additional responsibility items (rows) that involve neither called nor offered Service Operations can be included to clarify some key functional features of the Service Domain.

The sequencing of the rows within the general responsibility categories follows the typical flow of processing that might be expected when an offered Service Operation is called where appropriate, but this sequencing is informal.

The fields that are used to describe the responsibility items are:

<i>Capability/Delegation</i>	is a flag that reflects whether this row describes responsibilities covered by internal functionality or that is delegated to another Service Domain
<i>Responsibility Name</i>	is a short form descriptive name of the action covered
<i>Responsibility Description</i>	is a free form description of the action covered inasmuch detail as necessary
<i>Delegated Service Domain</i>	is only completed for responsibility items that delegate activity and this names that target Service Domain
<i>Delegated Service Description</i>	is a free form description of the delegated service, including any needed clarification of the purpose for the call
<i>Capability/Delegation</i>	is a flag that reflects whether this row describes responsibilities covered by internal functionality or that is delegated to another Service Domain

## BIAN Service Domain Template - Responsibility Items

Service Domain Name						
Description An outline of the business role taken from the V1.5 definition, expanded as necessary			Focus Object The focus object	Reference Object(s) Reference object(s)	URL	
Responsibility Category						
Initialize & Register						
Invocation & Execution						
Maintain & Analyze						
Report & Notify						

Functionality – was covered by Level 1 The functional responsibilities of the Service Domain are listed in the four responsibility categories. Sections include: <ul style="list-style-type: none"><li>• <b>Description</b> – free form text that describes the functional responsibility</li><li>• <b>Semantic Name</b> – structured fields that define the responsibility semantically</li><li>• <b>Delegated Function</b> – if appropriate each single delegated service and its associated Service Domain</li><li>• <b>URL</b> – pointers to any relevant documentation specific to the responsibility</li></ul>		Services – was covered by Level 2 The service operations are listed under the three applicable responsibility categories (no external access is provided for Maintain & Analyze). Sections include: <ul style="list-style-type: none"><li>• <b>Description</b> – free form text that describes the Service Operation</li><li>• <b>Semantic Name</b> – structured fields that define the service operation semantically</li><li>• <b>Input Parameters</b> – examples of up to six different types of input parameters</li><li>• <b>Output Parameters</b> – examples of up to six different types of output parameters</li><li>• <b>URL</b> – pointers to relevant documentation</li></ul>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

Capability/Delegation	Responsibility Name	Responsibility Description	Delegated Service Domain	Delegated Service Description	Delegated Service Operation
C or D indicates that the function is internal or that a service is invoked to complete the task	A structured name for the associated capability	A free form description of the item of functionality	An optional reference to a Service Domain if part of the function is delegated	A free form description of the functionality that is delegated by calling an service operation	Where available, the name of the called service operation

Figure 11 - The Responsibility Item Region Of The SD Template

### 3.1.3.4.4 Service Operation Details

The last region of the BIAN Service Domain template defines the offered Service Operations. These semantic descriptions represent the core of the interoperability standards that BIAN seeks to define. As they provide external access to the Service Domain they only apply to three of the four responsibility categories in the template:

- ✓ **Initialize & Register** – Service Operations called to establish a new record or occurrence for the Service Domain to handle

- ✓ **Invocation & Execution** – Service Operations to update an existing record, execute some suitable action against it or request that work is done by it
- ✓ **Maintain & Analyze** – as this is internally scheduled work there are no offered Service Operations (there can be delegated Service Operations)
- ✓ **Report & Notify** – Service Operations requesting ad-hoc or scheduled reports and subscribing to notifications for one or some collection of active focus object records handled by the Service Domain

## BIAN Service Domain Template - Service Operations

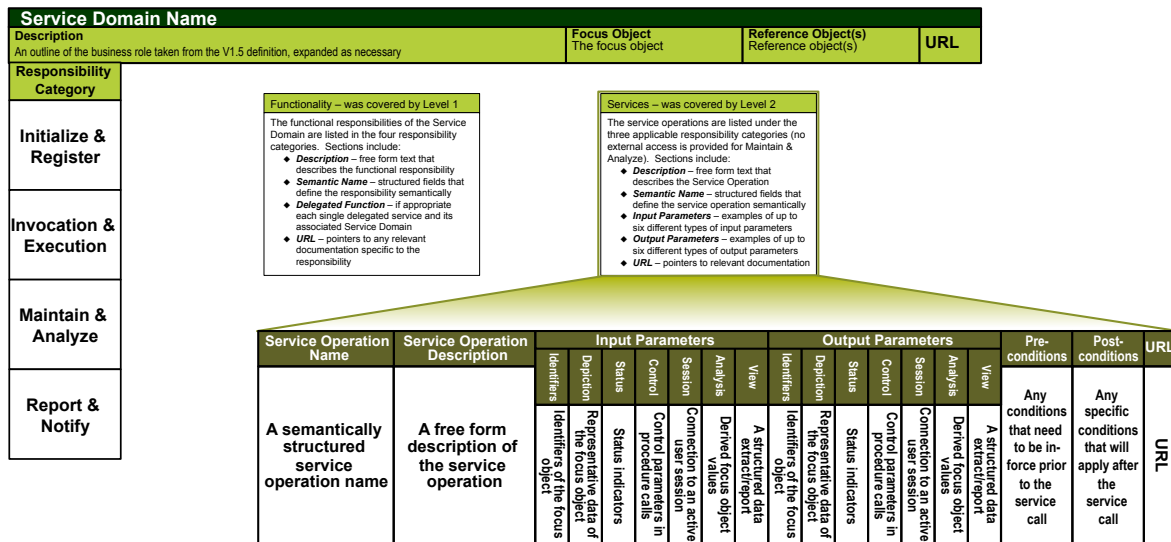


Figure 12 - The Service Operation Region Of The SD Template

The fields used to define the Service Operations in the BIAN Service Domain template are:

<i>Service Operation Name</i>	descriptive name, conforming to formal BIAN naming conventions
<i>Service Operation</i>	a free format description of the purpose and implied result of a call to the Service Operation
<i>Input Parameters</i>	lists of the business information provided as inputs to the Service Operation, organized in seven general categories (described below)
<i>Output Parameters</i>	lists of the output business information in the same seven categories
<i>Pre-conditions</i>	any notable circumstances that need to prevail before calling the Service Operation
<i>Post-conditions</i>	any notable circumstances that are likely to prevail or essential follow up tasks following on from the Service Operation call
<i>URL</i>	is a field where pointers to any supporting documentation of interest is captured (typically in the public domain)

The Service Operations typically act on one or more Service Domain focus object records. The seven input and output parameter types are:

<i>Identifiers</i>	information used to isolate one or a selection of focus object records
--------------------	------------------------------------------------------------------------

<b>Depiction</b>	information that represents details maintained in the focus object record
<b>State</b>	information that reflects what state a focus object record is in, typically associated with stages in its lifecycle or transitional states associated with some aspect of processing
<b>Control</b>	covers parameters that determine how some selected action is to be performed to or by the Service Domain and its focus object
<b>Session</b>	is rather specific information needed to support the transfer of access to an active customer contact between Service Domains for the few Service Domains that support customer interaction
<b>Analysis</b>	is derived information calculated by the Service Domain and associated with one or more of its focus object records
<b>View</b>	is a formatted extract of focus object and/or analytical information in a structured report

A key aspect of the semantic specification of the Service Operation is the definition of its input and output parameters. These should be complete enough to define the key business information content and the terms used must comply with the evolving BIAN business vocabulary. More comprehensive guidelines will be developed as BIAN members gain experience with creating and applying specification content.

The complete Service Domain template shows the four responsibility item categories and the association between offered Service Operations and the internal functionality that is needed to process them.

Service Domain Name													
Description								Focus Object		Reference Object(s)			
An outline of the business role taken from the V1.5 definition, expanded as necessary								The focus object		Reference Objects			
Responsibility Category	Capability/Delegation	Responsibility Name	Responsibility Description	Delegated Service Domain	Delegated Service Description	Delegated Service Operation	Service Operation Name	Service Operation Description	Input Parameters				URL
									Identifiers	Status	Control	Analysis	
Initialize & Register	C	Process New Record Request	The primary activity setting up a new record	NA	NA	NA	Create Focus Object	A service used to set up a new instance	ID	Active	NA	NA	URL
	M/C	Process New Record Request	The primary activity setting up a new record	NA	NA	NA	Create Focus Object	A service used to set up a new instance	ID	Active	NA	NA	URL
	D	Request Customer Account	Arrange for the set-up of a supporting customer account for the facility	Customer Account	Set up a new account	Create customer account							
	C	Check for possible duplicates	Check history to see if a similar application has been processed/rejected	NA	NA	NA							
Invocation & Execution	C	Process Deposit Request	Capture and verify a deposit request	NA	NA	NA	Process Deposit	A service used to make a deposit to the account	ID				URL
	C	Process Withdrawal Request	Capture and verify a withdrawal request	NA	NA	NA	Process Payment	A service used to make a payment from the account	ID				URL
	M/C	Process Transfer Request	Capture and verify a payment request	NA	NA	NA	Process Transfer	A service used to make a transfer from the account	ID				URL
	D	Debit the account	Check and extract amount from the balance	Customer Account	Make a withdrawal	Withdrawal to account							
	D	Transfer payment	Make an inter-account transfer	Payment Execution	Make a transfer	Payment request							
Maintain & Analyze	D	Calculate daily interest	Get the account activity and calculate and apply end of day interest	Customer Account	Get account statement for the day	Report on account							
	D	Calculate and apply fees	Determine and apply fees to the current account	Customer Account	Post fees to the account	Post charges							
	D	Execute Standing Orders	Make payments corresponding to standing orders	Payment Execution	Process Payments for Standing Orders	Payment execution							
	C	Maintain account tax position	Update the tax accumulated positions for the account	NA	NA	NA							
Report & Notify	C	Query Customer Account	Identify a customer account matching given details	NA	NA	NA	Query Account	A service used to identify/query a customer account	ID	Reference			URL
	C	Read Customer Account	Assemble and respond to a read account request	NA	NA	NA	Read Account	A service used to read a customer account	ID				URL
	C	Get Customer Account Report	Assemble a report covering many customer accounts	NA	NA	NA	Read Account Report	A service used to read a report covering customer accounts	ID				URL
	C	Notify Account Update	Notify subscribers on the event of an account change	NA	NA	NA							

Figure 13 - A Complete Service Domain Template

### 3.1.3.5 Message Level Mapping

A final consideration for the Service Domain template is the mapping from Service Operations to available standard messages. Often the semantic Service Operation details are at a higher level of detail than the field level message specifications. However for some Service Operations a good match can be made. The primary target for this mapping is the content in the ISO 20022 Repository.

The mappings are handled on a case-by-case basis for the Service Domains involved. At this stage there is no mapping detail maintained in the standard BIAN Service Domain template

### 3.1.3.6 Using the BIAN Service Domain Template

The BIAN Service Domain template will usually be referenced in the broader context of one or more Business Scenarios that invoke it. The template can be used to assess existing systems, for package selection and as a high-level design blueprint for new application development:

- *As a high level functional checklist* – the responsibility item list provides a functional checklist and the focus object an overview of the key business information processed by the Service Domain that can be compared with existing systems to identify shortfalls or as requirements for package selection and development. The delegated responsibility items also clarify functionality that should be sourced elsewhere rather than handled internally. Good service-oriented design reduces fragmentation and redundancy by partitioning functionality into Service Domains each with a narrow/specialized business focus that can be widely referenced.
- *To provide an industry standard service boundary* – the Service Domain Service Operations provide the semantic definition of the interface boundary for supporting applications. These can be used to re-align and rationalize existing application interfaces (a procedure sometimes called service enablement) and to specify the interface requirements for new package integration or custom development

The BIAN Service Domain template contains the semantic Service Operation designs that represent the industry interoperability standards that BIAN seeks to establish.

### 3.1.4 Data Repository Tool

For 2.0, the workgroups used the PowerPoint and Excel templates for their content development. The data was then keyed into a data repository tool which embeds the BIAN Metamodel into its structure. Going forward, content development will be done directly in the Data Repository Tool.

Use of the tool will increase the speed and quality of releases as workgroups no longer need to search for service domains, but can access them immediately, and changes are automatically visible to all.

The procedures and training materials for the Data Repository Tool are still being developed. The fields are identical to the spreadsheet templates, so the field references and explanations are the same. Here is a picture from the tool for reference.

## Business Scenario View in Data Repository Tool

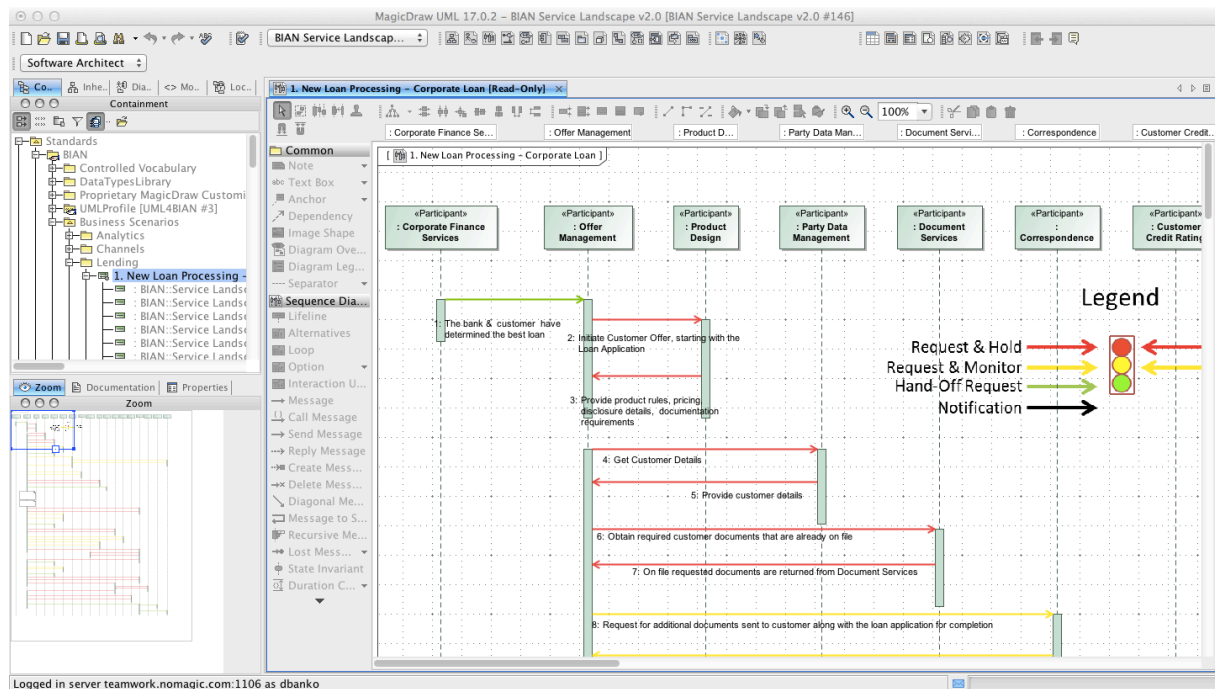


Figure 14 – Business Scenario View in Data Repository Tool

### 3.1.5 The BIAN Metamodel and Business Vocabulary

The design elements of the BIAN SOA Design Framework are supported and enabled by some key specifications and utilities. These include:

1. The BIAN Metamodel
2. The BIAN Business Vocabulary
3. The BIAN How to Design Guide
4. The BIAN general publication tools and facilities.

*The BIAN Metamodel* – the BIAN Metamodel captures all BIAN's design and specification concepts in a UML model. The model is fully documented elsewhere, points of note include:

- ISO 20022 compliant – the BIAN Metamodel is an extension of the Metamodel defined by the ISO 20022 financial industry standard. The BIAN Metamodel adds features to support BIAN's unique semantic designs of the Service Domain and associated structures. By virtue of the fact that the BIAN Metamodel extends the ISO 20022 Metamodel, BIAN adopts the ISO 20022 structures for lower-level design elements, avoiding the need to create additional, potentially conflicting definitions.
- Alignment for Tooling – by ensuring all BIAN design concepts align to the BIAN Metamodel, options for providing tooling support are enabled for the range of Service Domain related designs and standards, including the BIAN Business Vocabulary.
- Cross Standard Mapping – the BIAN Metamodel defines structures that support mapping between different vocabularies (both standard and proprietary). This feature is revisited with the outline of the BIAN Business Vocabulary next in this section.

*The BIAN Business Vocabulary (in-progress)* – In summary BIAN recognizes that a comprehensive business vocabulary is needed to support the semantic definition of Service Domains and Service Operations as well as in all naming conventions. BIAN's policy is to adopt established standards where available but at this time there is no single comprehensive business vocabulary covering the industry.

BIAN hopes to help the evolution of an industry standard business vocabulary through a facility that allows BIAN to map to prevailing terms and introduce new terms. The semantic definitions of terms in the vocabulary facility are used to track and display the similarities and differences between equivalent terms used in different existing vocabularies.

*The BIAN How to Design Guide* – this document, provides an overview of the BIAN designs, how they are used and how they are developed for reference.

*The BIAN general publication tools and facilities* – BIAN operates a range of web-based team working and publication facilities for use by its members and the general community. These facilities are constantly being refined as new design and publications tools are made available. Refer to [www.bian.org](http://www.bian.org).

## 3.2 Intended Uses of the BIAN SOA Design Framework

This section covers the approaches used to apply BIAN designs. One addresses the use of BIAN interoperability standards in the narrow context of one or more specific business processes. It considers deployments in different technical environments likely to prevail at banks. The second approach uses BIAN designs to develop a ‘blueprint’ of an enterprise that can be used as a framework for many types of assessment and planning purposes.

### 3.2.1 Applying BIAN Designs for Targeted Solutions

This section on Targeted Solutions describes how BIAN designs can be applied to a specific activity or narrow area of operations as might be represented using a single or small number of conventional business processes. It takes account of different application environments and architectures that might prevail at the Bank. The general approach is described in the following steps:

- 3.2.1.1 – *Select and Amend the Business Scenario* – the functional scope for the assessment is defined using one or more business scenarios
- 3.2.1.2 – *Define the Target State Requirements* – the basic business scenario and generic BIAN Service Domains are adapted to match the specific location and their design details extracted to create a target state requirements definition
- 3.2.1.3 – *Mapping Existing Systems to the Target State Business Scenario* - existing systems are related to the amended business scenario, including capturing any key non-functional considerations
- 3.2.1.4 – *Shortfall Impact Analysis* – gaps, overlaps and misalignment issues are identified and their impact outlined
- 3.2.1.5 – *Solution Options* – options will depend on the legacy environment. General approaches are defined that involve service alignment, service enablement/wrapping, portfolio rationalization and new solution integration

Situations where a targeted solution approach is applicable (in anticipation of case studies from the BIAN membership as the evolving BIAN standards are adopted) are briefly described in 3.2.1.6 – *Example Business Uses*

The overarching reasons behind the use of Service Oriented Architectures (SOA) and designs are wide ranging. Some expected benefits and justifications that relate specifically to the BIAN Service Domain designs include:

- *Functional specialization & optimization* – each Service Domain fulfills a narrowly defined business purpose, its internal workings can therefore be optimized to support its specific role avoiding the complexity and inevitable compromises associated with broader scoped (typically process based) solutions
- *Reduced fragmentation & duplication* – each Service Domain delegates any activities that are not directly associated with the execution of its own specific business purpose (i.e. supporting the full lifecycle of its focus object instances). By externalizing these requirements, duplicated functionality and the fragmentation of business information is minimized



- *Increased re-use and consistency* – All business activity can be supported by some combination of the standard collection of general BIAN Service Domains. As each Service Domain has a unique and discrete purpose it acts as the single 'go to' place for all callers to access its capabilities for any business situation that might need them.

Fortunately, experience has shown that a migration towards service based application partitioning can be handled incrementally with good design. Existing application portfolios typically include high levels of redundancy with overlapping processes and the associated duplication and fragmentation of business information. The integrity and performance of these application portfolios can be improved with the selective imposition of discrete, well-focused functional partitions that are service enabled. The BIAN approach to Targeted Solutions leverages this powerful property of SOA.

By adopting BIAN generic Service Domain boundaries and their standard Service Operations it is intended that the benefits of service enablement are optimized, that interoperability within and between organizations is improved and that in time aspects of the designs and solutions developed can be more easily redeployed.

The approach is detailed in the next six sections.

### **3.2.1.1 Select and Amend the Business Scenario**

One or more business scenarios are selected that define the primary activities of the target area. BIAN has an expanding inventory of general business scenarios that are used in the development of its Service Domain designs. As a result where BIAN Service Domain designs are available there should be associated candidate BIAN Business Scenarios for review.

If available, the BIAN Business Scenarios are still likely to require amendment to reflect the target state requirements of the organization. This can be to add missing steps/service calls, to change the sequence or expand the rules governing Service Operation calls within the scenarios. If no business scenario is available, the BIAN Service Domain definitions can be used to assemble new Business Scenario using available scenarios as a guide. It is important when amending or creating a business scenario that the business purpose of each involved Service Domain is not contradicted.

Finally, walk through the completed business scenarios with business practitioners to confirm they collectively cover all the key workings of the target area and if possible with application architects familiar with the BIAN specifications to confirm the Service Domains have been correctly interpreted.

### **3.2.1.2 Define the Target State Requirements**

The specifications of the BIAN Service Domains referenced in the Business Scenarios represent prevailing, general business functionality. The responsibility items and their Service Operation descriptions in the Service Domain template are functional checklists. These, when combined with the Business Scenario, provide a definition of the target state requirements.

Most of BIAN Service Domains should map directly to a corresponding business capability at a location. There will be cases where the Service Domain is either too narrowly or too broadly scoped. For example, BIAN currently defines a single Service Domain to cover the accounting records for customers. It is possible that a bank may wish to further specify customer accounting activity by market segment. In this case, there would need to be multiple versions of the same BIAN Service Domain, each adapted as necessary to match the unique needs for its specific segment. One of these more specialized Service Domains is likely to be involved in the target Business Scenario(s).

Another possibility is that BIAN has defined two or more generic Service Domains but a bank wishes to treat them as a single entity. For example BIAN has defined product fulfilment Service Domains for both consumer loans and consumer deposits. If a single service center is required combining loans and deposits the responsibilities and Service Operations for the two Service Domains are merged into a single design entity and any redundancy removed.

When defining the target state, there will also often be unique local requirements that need to be added to the generic BIAN designs. Such additions typically reflect procedural constraints or features imposed by legacy systems and procedures but can also be market differentiating competitive

behaviors that have yet to be adopted generally by the industry (and so are not reflected in the generic BIAN specification). Part 4 of this guide includes more detail about the scoping techniques used to define a Service Domain and these should be referenced before making extensive changes to the scope and content of a BIAN Service Domain.

Once each Service Domain of the business scenario has been reviewed and amended as necessary, the Service Domain templates collectively provide high-level functional specifications that can be used to define a target state requirements specification for the business scenario. This target state specification is used for assessing existing systems or as the initial design for new development. The content of the templates define the target state requirements as follows:

1. The responsibilities of the Service Domains provide a high level checklist of the major functions that should be performed by the supporting systems in well-bounded partitions. This includes identifying the delegated access between Service Domains and associated systems
2. The Service Operations provide high level specifications of the interfaces and underlying service and message interactions
3. The focus object and associated object model provides a semantic outline of the business information governed by the Service Domain and corresponding information accessed from other Service Domains

### 3.2.1.3 Mapping Existing Systems to the Target State Business Scenario

The target state is a framework against which the coverage 'footprint' and quality of this coverage for existing systems (and candidate packages) can be mapped. The purpose of the mapping is to expose shortfalls in system coverage. These shortfalls fall into three main categories:

1. *Gaps* – key functions or Service Operations are not adequately supported
2. *Duplication* – two or more existing systems compete to provide the same capability when one would suffice
3. *Over-Extension* – a system suited to support the working of one Service Domain is extended to support two or more, potentially compromising performance and obscuring Service Operations

In many cases the existing systems will map neatly to a single Service Domain. It is also possible that smaller/more specialized systems will support some sub-set of a Service Domain's functionality. As long as a combination of well-integrated systems can be identified to cover the workings of the Service Domain with no overlap these systems collectively map to the framework. Functionally mapped systems can then be evaluated for their 'service alignment.' Service alignment involves two main adaptations:

*Supporting Service Operations* – existing interfaces may need to be improved to generalize and service enable them so that they can be called by any eligible party. Where multiple non-overlapping systems support a Service Domain this may involve additional integration to support some 'shared' or co-dependent Service Operations

*Externalizing duplicated functionality* – a more involved adaptation involves extracting functionality that should be supported externally through delegation. Before functions can be externalized there clearly needs to be a suitable service provider to which internal connections can be redirected

A more problematic mapping is when a single system maps to part of two or more Service Domains. When this is the case, it is possible that there will be some level of operational compromise because a single system may not be architected in a way that fully supports the potentially conflicting needs of two or more Service Domains. It may also not be possible to expose the full range of operational services for all mapped Service Domains if the system has a monolithic (non-modular) internal structure.

Finally the system mapping can reveal where two or more systems cover the same functions within a Service Domain or for complete Service Domains. This is often the case where a bank has siloed production capabilities where different systems have been built over time to fulfil the same function for different products or locations.



### 3.2.1.3.1 *Non-functional Properties*

When current systems are mapped to the target state framework it can be useful to capture any extreme technical and operational requirements against each Service Domain. These needs may provide insights into the selection of candidate systems later in the analysis. Topics include above normal concerns for security, performance, availability and any need for advanced/specialist technology such as for analytics and communications.

(The definition of these requirements falls outside the current scope of BIAN's activities but the topic is noted here for completeness)

### 3.2.1.4 Shortfall Impact Analysis

Once current systems have been mapped to the target state framework the impact of the identified shortfalls need to be assessed. Whether the shortfall is a gap, duplication or over-extension, the impact on business performance should determine what priority is given to any corrective action. As service enablement can be achieved incrementally it is usually important to focus first on shortfalls where improvements are both feasible and the business impacts are greatest.

The assessment of business impact is going to be specific to the organization but some general guidelines can be given as to the likely types of benefits corresponding to the three categories of shortfall already defined:

1. *Gaps* – missing or limited systems support for functions and Service Operations can be associated with increased operating costs, inefficient use of resources, ineffective or constrained business performance, poor decision making and increased incidence of preventable errors. “Before and after” simulations can be used to associate financial measurements with the areas of business operation that are not adequately supported
2. *Duplication* – introduces obvious additional support costs for maintaining parallel systems and operational capabilities. Duplicated systems can also lead to data and process integrity/consistency issues with an associated business impact. Though this financial impact can be significant, it is important to confirm that it adequately offsets the potentially significant costs associated with rationalizing duplicated systems. A fairly detailed assessment of requirements should be undertaken as activity that appears similar at a high level can turn out to be necessarily quite different at lower levels of detail. The costs of enhancing one system to take over the role of many and the subsequent migration effort can outweigh the business benefits. Engineering a systematic phased migration that allows obsolete systems to ‘atrophy’ gracefully can be the best approach
3. *Over-Extension* – many of the larger legacy systems will span multiple Service Domains. If the legacy systems have a modular design it can be fairly easy to expose the discrete Service Operations for the different Service Domain partitions and any operational compromises arising from having a shared technical architecture can be tolerated. Problems arise when the architecture is monolithic and enhancements to support the different Service Operations are hard or impossible to engineer. In such cases the business impact typically reflects the gaps that result from the lack of Service Operations and from functionality that can't be added to the legacy code base. The business impact is again estimated using before and after simulations.

### 3.2.1.5 Solution Options

The approach to addressing shortfalls will be highly specific to the prevailing situation at any bank. In general terms the BIAN designs provide a framework for aligning systems in a service oriented architecture with the anticipated benefits outlined at the head of this section. These alignments are:

## How Legacy Applications can be Related to Service Domains (existing interfaces are 'aligned' to service operations)

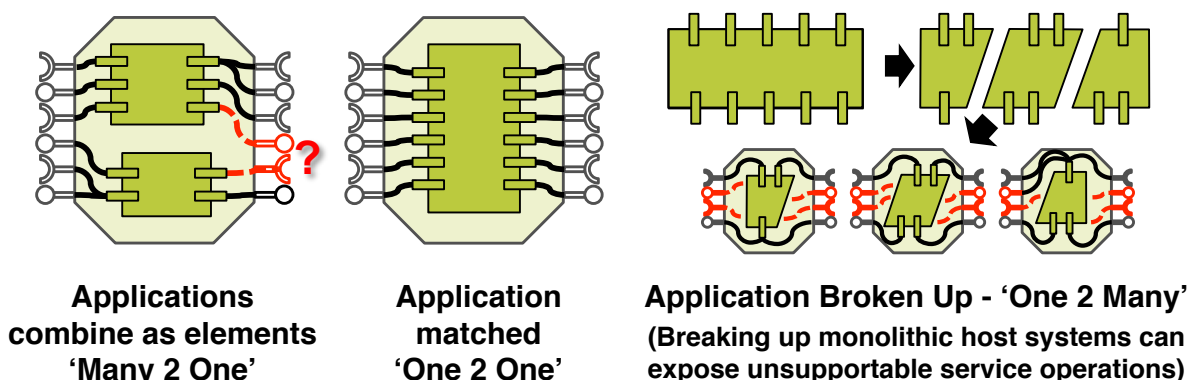


Figure 15 - Three Mappings: Elements, One-2-One, One-2-Many

When many non-overlapping applications combine 'many 2 one' to support a single Service Domain it is necessary to ensure adequate integration exists to support all Service Operations such as those associated with analysis and reporting in addition to any changes needed to service enable the systems. For a 'one to one' mapping any changes will relate to appropriate 'externalization' and 'service enablement' (see Section 3.2.2.3). For a 'monolithic' application that supports many Service Domains the solution needs to modularize and service enable the application such that it can appear to support each area independently to an outside caller.

The technical approach to achieve these different alignments differs for the various technical environments that might prevail at a bank. In general terms possible migration techniques include:

1. For monolithic legacy applications and packages
  - The mapping should define logical modules within the scope of the legacy applications, with the corresponding Service Operations
  - The legacy application may be sufficiently modular and easily enhanced to be re-worked to support the service enabled design
  - Where the functionality is constrained a wrapping framework can be applied to augment the legacy application, typically directly accessing core data structures and assembling and managing a service based external interface
  - Functionality can also be suppressed for Service Domain areas that are not well covered within the monolithic system and alternate external approaches developed and integrated.

For many legacy applications there will be significant compromises when attempting to align to the target service oriented architecture. Centralized databases, involved and interconnected procedural software and unwieldy development and testing facilities will all limit the extent to which a monolithic legacy application can be made to mimic a collection of loose coupled service centers.

2. For highly fragmented conglomerated systems typically mixing host transaction processing (TP), client server and more recent web based platforms
  - Major TP host systems can be handled in a similar manner to that described above for monolithic legacy systems
  - Server based applications are likely to align well to the Service Domain boundaries requiring manageable efforts for any required externalization and service enablement
  - The flexible sever environment is also well suited to support legacy wrapping and additional package integration or development as may be required
  - Any "fat client" applications are likely to embed logic that should be relocated within server based Service Domain aligned applications
3. For green-field service based developments the target state framework defines a high level blueprint for configuring and service enabling commercial packages and for outlining the major modules and service interfaces for custom development

### 3.2.1.6 Example Business Uses

BIAN will seek to identify and document member case studies that provide practical examples and lessons learned from using BIAN standards. The types of case studies that use some aspect of the targeted solution approach include:

#### **Assessing or Implementing a Point Solution**

A member case study would follow the targeted solution approach for a narrowly defined aspect of the business, covered by one or more business scenarios and a small number of associated Service Domains. This is shown in the SunGard Case Study in Appendix 2.

#### **Product Launch**

A member case study would cover a product launch in a similar manner to a Point Solution, involving a small number of business scenarios. In addition, the broader Service Landscape identifies interested, influencing and impacted areas of the business for broader launch planning and coordination purposes. This is shown in the SunGard Case Study in Appendix 2.

#### **Core Systems Repurposing**

A member case study covering Core or Legacy repurposing would relate to the first solution option described in the previous sub-section (Section 3.2.1.5 point 1) and would be a good confirmation of the ability to use BIAN partitions and designs as a blueprint for a legacy system wrapping solution.

#### **Application Portfolio Rationalization**

A member case study would involve matching applications to the service domain capabilities to reveal gaps, duplication and misaligned applications in the overall portfolio, then using the mapping to do like-for-like comparisons of competing applications with respect to their coverage of individual service domains. This is shown in the Scotiabank Case Study in Appendix 2.

#### **Mergers & Acquisitions**

Similar to Application Portfolio Rationalization, a member case study would involve matching applications from different organizations coming together to the service domain capabilities to reveal gaps, duplication and misaligned applications in the overall portfolio, then using the mapping to do like-for-like comparisons of competing applications within the organizations with respect to their coverage of individual service domains.

#### **Vendor Solution Alignment**

A member case study covering a vendor solution integration where the vendor offering corresponds to a targeted solution and the Service Domain and Semantic Service designs are used to develop a high level functional checklist and interface definitions. The target solution will be used differently by banks and software vendors:

*For the Bank* – it is used to define required functional content coverage and interfacing. Compliance with the evolving standards can also be a measure of the level of commitment a supplier has to standards.

*For the Software Vendor* – solutions that align to the generic BIAN Service Domains can be consistently mapped to requirements for multiple deployments, enabling better solution re-use and ease of integration. This is shown in the SunGard Case Study in Appendix 2.

#### **Investment Planning**

A member case study would involve using the Service Landscape as an investment planning framework. Different service domains can be evaluated in different ways to provide a backdrop for planning and prioritization. For example, costs can be allocated across the landscape to expose the relatively high cost areas where investment to improve efficiency could have the highest impact.

#### **Outsourcing/In-sourcing**

BIAN Service Domains can in practice be sourced externally since Service Operation interactions can be supported. A member case study would use the Service Operations as

the basis for supporting an outsourced arrangement. Outsourcing decisions can be evaluated by considering individual and groups of Service Domains across the landscape as candidates for flexible sourcing approaches.

## 3.2.2 Applying BIAN Designs for Enterprise Planning

This section describes the use of the BIAN SOA Framework for enterprise or major division management. It is set out in three sub-sections:

*3.2.2.1 – Building a Business Blueprint* – the BIAN Service Landscape is a reference structure with one and only one of each of the general BIAN Service Domains – this model needs to be amended to map to the current or future layout of any specific Enterprise

*3.2.2.2 – Attributing a Business Blueprint* – once a business blueprint has been developed for an organization it can be used as a framework for structuring current and future state analysis

*3.2.2.3 – Examples of Blueprint Based Planning & Analysis* – some candidate examples for which BIAN would like to develop case studies

### 3.2.2.1 Building a Business Blueprint

The BIAN Service Landscape is a reference framework for classifying and organizing the complete collection of BIAN Service Domains. Each Service Domain is a general design of a discrete capability and the Service Domains collectively are intended to cover all aspects of the Banking industry. The BIAN Service Landscape is not a general model of a Bank; it is more a structured 'library' index that contains one of each of all the possible building blocks that might be needed to build any bank.

To assemble a model of a particular financial institution or its business 'blueprint' using BIAN Service Domains there are several considerations to be handled, in simple sequence they include:

1. *Define the Blueprint Layout* – a wide range of formats can be used to layout the Service Domain elements to make up a blueprint. One example is described later in this section but the layout of the blueprint can be highly specific to any one enterprise
2. *Select Service Domains* – the BIAN Service Landscape includes all possible Service Domains (those identified so far). A subset of these general Service Domains typically will be used within any one financial institution. For example BIAN seeks to have Service Domains covering all possible products and most banks will specialize on a sub-set of products. Also depending on the active products, market segments and channels many other general Service Domains may be excluded
3. *Determine Organizational Duplication* – in most enterprises many activities may be repeated in different physical locations, matching the management command and control layout and geographic 'footprint' of the enterprise
4. *Determine Functional Variations* – The general BIAN design content will usually be significantly extended in implementation. For most Service Domains this will result in a single amended Service Domain but there will be certain Service Domains for which multiple versions will be required in the blueprint. This is commonly the case where the same activity is repeated in different market segments, lines of business and/or geopolitical climates

The business blueprint for an enterprise provides a concise 'bank on a page' representation. To be compatible with the BIAN standards, all that is necessary is that all the elements of the blueprint correspond to BIAN Service Domains (either one to one, or many to one). The four steps above are next developed using an example to clarify the approach. Note the example is described as if the model is being defined to match the current state of the enterprise for simplicity. Depending on the specific situation the blueprint might be designed to reflect some future state model if a significant restructuring is expected.

#### 3.2.2.1.1 An Example Layout

The BIAN Service Landscape uses a two-level hierarchical classification of Service Domains using Business Areas and Business Domains. In Section 2.2 a variation of this layout was shown with the selection criteria more closely aligned to an organizational classification. However this example layout still limits the resulting framework to having one of each general BIAN Service Domain.

An enhancement of this approach can be used to define the general layout of the blueprint that allows for Service Domain selections, duplications and variations. Two common reasons for an organization having parallel capabilities are that it is active in multiple segments and/or multiple geopolitical locations. This structure will usually be reflected in the divisional organization of the enterprise.

A divisional structure will typically combine several major organizational components:

- Central (and perhaps regional) executive management and control units
- Enterprise-wide business functions (e.g. Finance, Systems, HR)
- Central shared services and operations (e.g. Document Archive, Billing & Payments, Accounting)
- Segment and geographic specific product delivery and relationship management (e.g. Product Fulfillment, Customer Relationship Management)

These (or similar) categories can be used to define the primary business areas of the blueprint where each area will typically align to an established (or target) 'Division' of the enterprise:

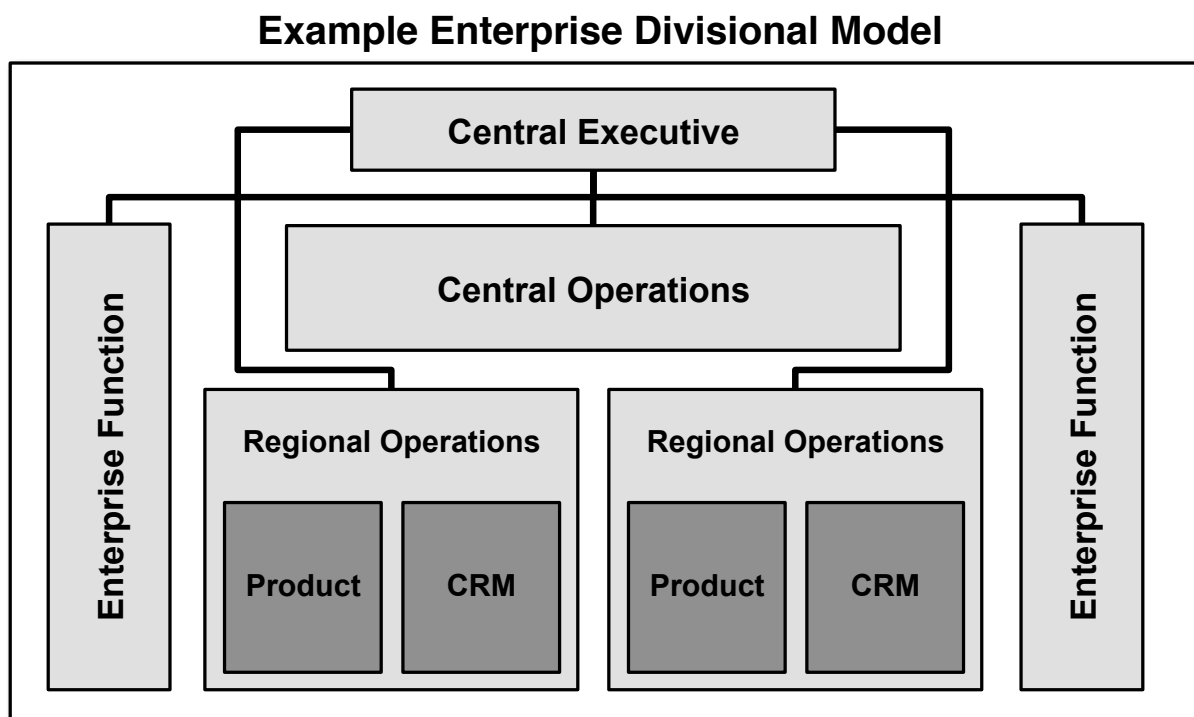


Figure 16 - An Enterprise Divisional Business Area Model

The Business Areas can now be sub classified into their constituent Business Domains. These partitions typically relate to existing (or target) Business Units with broadly defined business responsibilities similar to those seen in the example in Section 3.1.1.

Required organizational duplication should be clear at this stage as many Business Domains may be duplicated in different Business Areas (usually with regional or segment specific business units). At this stage of the blueprint definition some interesting decisions can be made about positioning business activity in central/enterprise divisions or repeating it in multiple segment and/or geographic locations. How these decisions are made is however beyond the scope of BIAN's proposed techniques but the result is well represented in the BIAN compliant enterprise blueprint.

### 3.2.2.1.2 *Selection, Duplication and Variations*

The overall structure of the blueprint sets out the high level organization or 'command & control' of the Enterprise with Business Areas and Business Domains. BIAN Service Domains can next be selected and any necessary organization duplication and functional variation reflected in the positioning of

Service Domains within the blueprint. These considerations were first described in discrete sequence but in practice this next level population of the blueprint with Service Domain aligned elements will address these in parallel.

Service Domains can be considered individually as they are extracted from their groupings in the BIAN Service Landscape. For each Service Domain the following decision making process can be applied to its selection, duplication, amendment and positioning:

1. Is the Service Domain's business role performed (anywhere) at the enterprise?
2. Is it (should it be) a central activity? If so there is probably a single location for it in one of the central Business Area Business Domains (enterprise management, enterprise services, central operations)
3. If on the other hand it is replicated and distributed across geographies and/or segments there will be repeated Business Domain locations for it in many Business Areas (Service Domains in this category typically cover some aspect of product/service fulfillment, servicing, sales or relationship management)
4. Finally for those Service Domain's that have been duplicated it should be determined whether the general BIAN Service Domain specification is sufficient, or whether one or more new adapted versions of the Service Domain is required in deployment

### Example Enterprise Divisional Model – with Service Domains

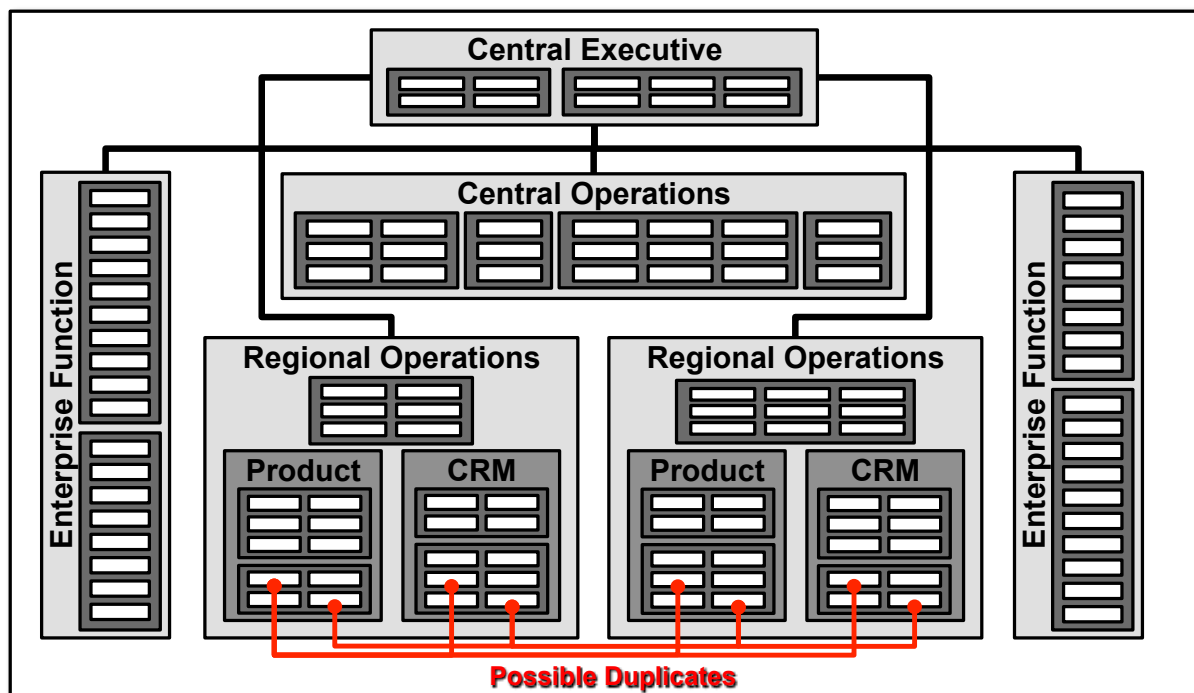


Figure 17 - An Enterprise Model With Service Domains Added

The Business Blueprint with its constituent Service Domain aligned building blocks provides a stable framework for general business analysis and planning.

#### 3.2.2.2 Attributing a Business Blueprint

A business blueprint built using BIAN Service Domain partitions is a particularly good framework for management analysis and planning for several reasons:

- *Stable over time* - the roles of the discrete business capabilities represented by Service Domain partitions do not vary over time. The way they might execute and the patterns of collaboration that might involve them will change as new practices and solutions evolve but their specific purpose and hence their position in the blueprint is unlikely to change. As a result a model built using these elements will itself be highly stable



- *Concise, implementation independent view* - beyond the broad-brush scope of business areas, the blueprint does not presume any specific organizational or technical approach. It can also be interpreted both by business and technical architects consistently bridging a design gap that often exists with other (incompatible) business and technology modeling approaches
- *Suited for overlays/attribution* – the elements that make up the model can be associated with resources and current or target performance measures to support a wide range of business performance assessments
- *High level design* – the blueprint can also act as the top level design of underlying organizational, procedural and information systems solutions providing a planning framework for targeting investment and solution development

The types of attributions and overlays that can be mapped to the blueprint are unlimited, indeed specific attributions can be developed to suit a particular type of management analysis. The types of attributions that can be considered fall into several general categories:

- *Resource mapping* – the ‘footprint’ of different types of resources can be associated with the Service Domain aligned elements of the blueprint. Sometimes this mapping can be carried down to the lower level specifications within the Service Domains themselves for more detailed comparisons and assessments. Resources that can be mapped include systems/applications, personnel and utilities such as technology platforms, buildings and equipment. Resource mappings can be used to highlight shortfalls such as gaps, duplication and misaligned resources
- *Performance Targets* – various measurements can be associated with the current or target performance of an element corresponding to a Service Domain. Using these measurements the blueprint becomes a dashboard, the stable nature of the framework being particularly suited to this use. Performance measures can be categorized as measuring the efficiency, effectiveness and/or flexibility/responsiveness of the Service Domain.
- *Properties* – properties can be associated with different aspects of the Service Domain element. Typical aspects include the people, procedural/operational and supporting systems. Properties that can be associated with these include:
  - *People properties* – staff numbers, skills/qualifications, authority, working arrangements, etc.
  - *Procedural/operational properties* – centralized/distributed, 24/7 operations, high security, highly automated, differentiating, etc.
  - *Systems* – high performance/availability, dependent on specialized technology, suited for STP, analytical/pattern detection, etc.

The main types of possible attributions are shown on the following simplified enterprise blueprint.

### Enterprise Divisional Model with Different Attribution Types

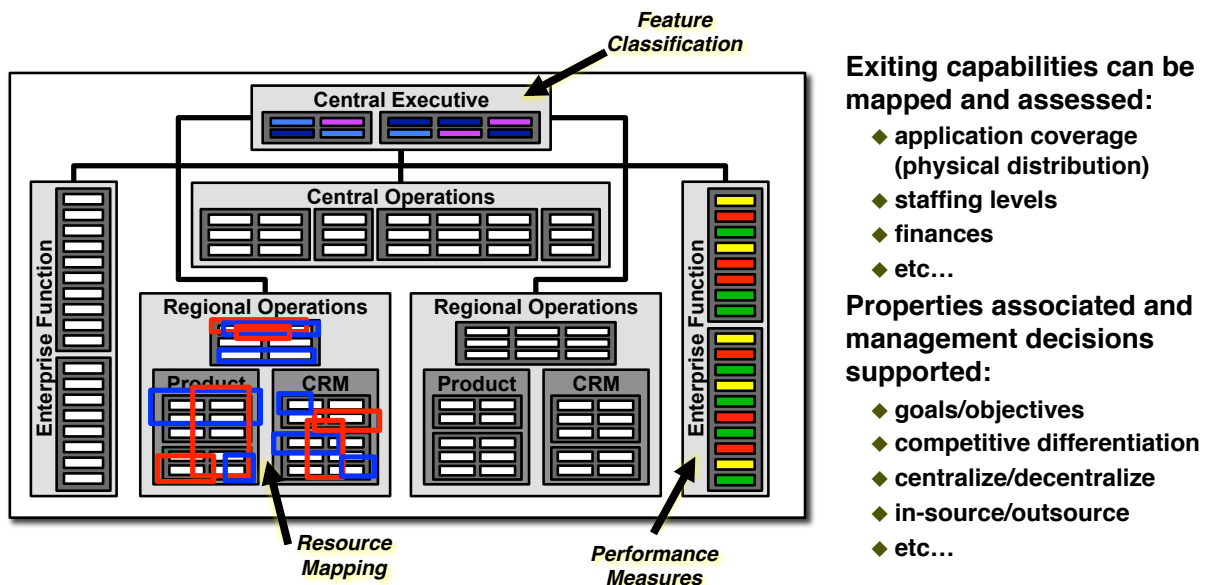


Figure 18 - Business Blueprint With Different Attributions

### 3.2.2.3 Examples of Blueprint Based Planning & Analysis

It is not BIAN's intention of defining the types of analysis and planning techniques that can leverage a business blueprint assembled from BIAN Service Domain designs. But BIAN is seeking to capture case studies of this type of use of BIAN designs from which the broader membership might benefit. Examples of suitable case studies include:

#### Application Portfolio Rationalization

The more detailed Service Domains responsibility checklists that underlie the blueprint can be used to do a mapping of the existing application portfolio to highlight shortfalls. Applications when matched to the Service Domain responsibility items reveal gaps, duplication and misalignment in the application portfolio. The mapping approach ensures a like-for-like comparison is made because overlapping applications are assessed with respect to their coverage of individual Service Domains.

#### Mergers & Acquisition

Mergers & Acquisitions is a case study similar to application portfolio rationalization but where specific emphasis is placed on comparing and contrasting competing applications brought together by the merger. Besides the mechanics of application portfolio rationalization, additional attributions can be used to help govern the overall merge process for example highlighting critical areas that need to be carefully managed or high cost areas that should be targeted early for rationalization

#### Investment Planning

The blueprint is a map of all the building blocks of the business that can be used as an investment planning framework – different Service Domains can be evaluated in different ways to provide a backdrop for targeting effort. For example, costs can be allocated across the landscape to expose the relatively high cost areas where investment to improve efficiency could have the highest impact.

Many other case studies can be defined and considered for inclusion in future versions of the guide.



## Part 4 – BIAN Design Techniques

---

This part of the guide describes the design concepts and supporting techniques used within the BIAN membership to develop the detailed Service Domain specifications. This includes the approach used by the individual BIAN teams to develop candidate Service Domain designs.

The audience for this part of the guide is BIAN members that can use this as a reference and training guide and for SOA Specialists that wish to better understand the adapted SOA theories underlying the BIAN approach.

This part of the guide is structured in four sections as follows:

*Section 4.1 – Concepts behind the BIAN Standards* – describes the design concepts that support the BIAN design techniques. In particular it goes into more detail than earlier in the guide as to how BIAN defines a specific type of service oriented architecture (SOA).

*Section 4.2 – The BIAN Service Domain Design and “2 Cycle” Development Approach* – explains the design techniques applied to partition a Service Domain in more detail. It also defines an internal working procedure that BIAN working groups use to develop candidate Service Domain specifications – the “2 Cycle” development approach

*Section 4.3 – The BIAN Metamodel & Business Vocabulary* – these items are fully documented in their own BIAN guides but some of the key features and considerations are repeated here for reference. In addition some specific fields in the design templates needed to support the business vocabulary are explained

*Section 4.4 – Positioning BIAN alongside Other Standards Organizations* – BIAN is positioned alongside some of the other key standards bodies active in the Financial Services industry

### 4.1 Concepts behind the BIAN Standards

A common goal of industry standards is that they are canonical, i.e. that they represent the typical form or approach that can be recognized and applied consistently. Furthermore, the procedures used to define standards need to be traceable and repeatable. Ideally the procedures should be specific enough such that if two appropriately skilled architects were to use them to model the same business activities independently they would produce the same result.

This level of precision is not always possible and so a full disclosure of the techniques and concepts applied is necessary so that where an arbitrary selection has been required it can be reviewed and the result amended if so desired. This is the case in the BIAN approach for rightsizing Service Domains where a bank can choose to revisit BIAN’s partitioning decisions as briefly described in Section 3.2.1.2.

In this spirit of full disclosure the remainder of this section describes three foundational concepts that underlie the BIAN semantic service standards:

*4.1.1 – BIAN Uses a Service Oriented Architecture (SOA)* – describes how BIAN defines its own specific version of a SOA

*4.1.2 – Business Model versus Software Model – An Issue of Scale* – makes a clear distinction between BIAN’s high level semantic designs and the far more detailed software designs that apply to support systems

*4.1.3 – Governing Service Domain Design Concepts* – lists some of the key properties and ‘checks’ applied to confirm Service Domain partitioning decisions

#### 4.1.1 BIAN Uses a Service Oriented Architecture (SOA)

To understand the BIAN design approach it is first necessary to explain the 'model view' inherent in the BIAN standards. A model view of anything is a simplified representation typically designed for some specific use or to provide some particular insight. For example a street atlas sets out a city in a format that is helpful to navigate its roads and pathways. The underground or rail map of the same city looks completely different because it is developed to support a different perspective or use.

There can be many model views of business activity; here two distinct model views are compared to clarify BIAN's approach to standards: process oriented and service oriented (called as a Service Oriented Architecture - SOA). BIAN uses its own version of a SOA and it is important to make this distinction clear because the BIAN standards can be misinterpreted if they are taken out of their correct model view context.

The service-oriented model view defines partitions that remain intact throughout their decomposition. The capability partition of the Service Domain defined at the top semantic level and its associated Service Operations can be traced through to low-level implementation designs.

## Service Domain Decomposition

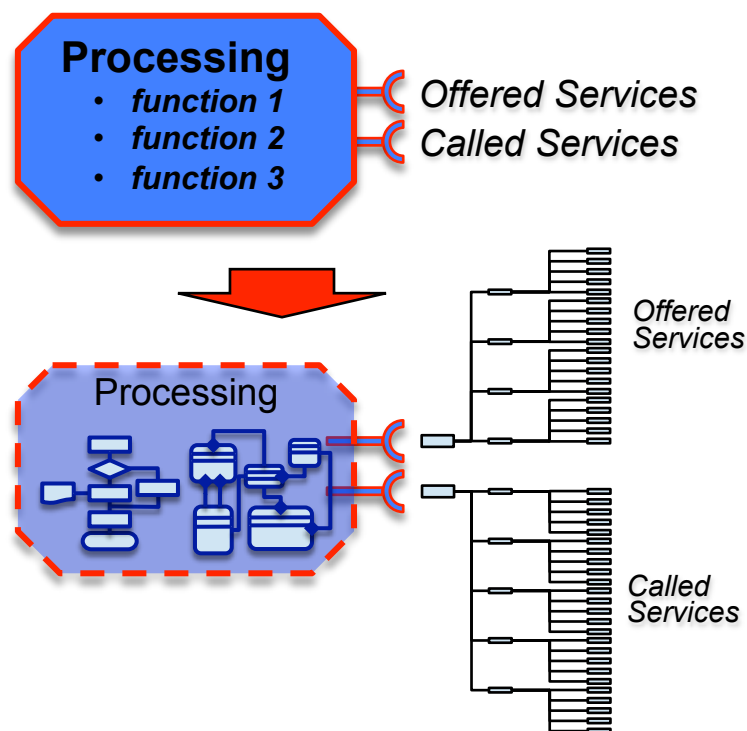
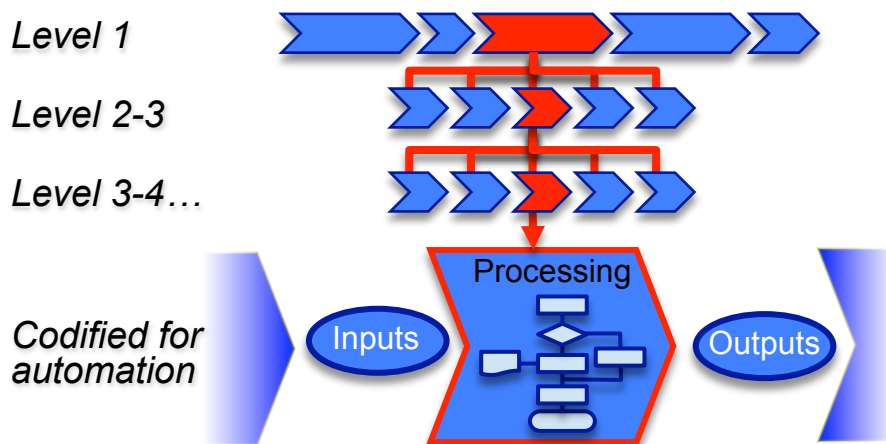


Figure 19 - Service Domain Decomposition

The translation to implementation level specifications addresses two distinct aspects of the design in parallel. For one the Service Domain's internal functionality is decomposed to fine grained specifications. For the other the high-level Service Operations are translated into more specific service exchanges with mapped message schemas that define the data elements they contain.

Service Domain partitions are implementation independent so as shown in the diagram the internal functional designs can adopt any suitable technical implementation approach. (Procedural and Object Oriented Analysis and Design (OOAD) models are shown schematically). The operational characteristics of the Service Domain will determine the best internal architecture. A high volume transactional Service Domain may be suited to process based design. A Service Domain with complex collaborative interactions may be suited to object oriented design. Since the Service Domain boundary is intact and all external interactions are service based the solution is compliant with SOA principles.

## Process Decomposition



*A business process is progressively decomposed to a level of granularity that can be represented as computer code*

Figure 20 - Process Decomposition

A process oriented view of business activity seeks to identify and represent the dependent series of linked tasks performed to get from one point or state to another point or state. A process model is usually developed from a top down perspective – first high level/big picture actions are defined with the conditional flow between them detailed in semantic terms. These actions and control flows are then progressively decomposed into ever finer/more specific detail, using the same action/conditional connection visualization (aka input/process/output).

When used to define and develop systems process decomposition provides a traceable path from a high level summary down through increasing layers of detail until the granularity of the actions and conditional steps are sufficiently detailed to be rendered in machine implementable logic.

Process modelling is a very powerful, flexible and proven design technique but the tracing of the process decomposition between the layers is largely informal. Partitions defined at the higher levels can be blurred or completely lost as detail is added. The high level partitions have no formal architectural significance at the lower implementation levels.

### 4.1.2 Business Model versus Software Model – An Issue of Scale

The BIAN Service Domain at the heart of the BIAN SOA model view represents a coarse grained operational capability; it is a business model element. A large bank with tens or thousands of employees can be modelled using a few hundred BIAN Service Domain partitions and all business activity represented as patterns of service collaborations between these Service Domains.

The same Bank may today operate hundreds of production system applications and any one of these can contain several millions of lines of code. The software of a production application, if procedural is likely to be made up of large number (thousands) of parameterized procedures/modules or if object oriented a similar numbers of nested software objects.

Though both procedural and object oriented software elements have many structural similarities to a BIAN SOA Service Domain, they represent fundamentally different design concepts not least being some 2-3 orders of magnitude apart in terms of scale. It is important not to misinterpret the BIAN SOA business designs as being representative of software process modules or software objects.

A BIAN SOA Service Domain partition equates more precisely to a (group of) operational function(s) of an enterprise, perhaps with many users (it can sometimes be highly automated in implementation) and this operating unit might operate one or more supporting production applications. The BIAN SOA is a

model representation of business and operational activity that helps to define optimal functional groups and the key exchanges between them.

This BIAN SOA business model view is however specifically aligned to service-based implementation. Where the functionality within a Service Domain and/or the nature of the information exchanges between Service Domains is suited to some level of computer based automation, supporting software implementation designs can be developed that align to the exact same structures.

To remain compliant with the BIAN standards production application boundaries should align to Service Domains. It is not essential for one application to match a single Service Domain; a Service Domain can contain multiple applications if they are well integrated. Issues can arise when an application spans Service Domain boundaries, performance can be compromised and Service Operations obscured as described in Section 3.2.2.

In summary: BIAN's SOA and the semantic Service Operation standards it contains define a business model view. They are structured specifically to enable supporting service based systems implementations. However the BIAN designs are agnostic of any specific underlying software implementation approach. BIAN's design artefacts should not be interpreted as any form of software design specification.

### 4.1.3 Governing Service Domain Design Concepts

BIAN's standards hinge on the correct definition of BIAN Service Domains. They need to be general/canonical so that their associated Service Operations define standard service boundaries. BIAN maintains governing concepts and definitions that seek to ensure the overall integrity of its design approach and the correct scoping of the Service Domains:

*Semantic Definition* – BIAN designs are defined at the natural language/semantic level. This means the designs describe business activity in general terms. The specificity of the designs aims to be sufficiently precise and comprehensive to define unambiguous behavior and responsibilities. However BIAN designs are not intended to be exhaustive – they define the mainstream activities that can be the subject of a generic/canonical design.

*Implementation Agnostic* – BIAN designs are implementation independent. BIAN designs can be extended and elements duplicated and amended to map to a specific implementation. BIAN designs do not presume any one specific underlying technology based implementation but act as a general blueprint for engineering well partitioned service oriented systems capabilities.

*SOA/Service Capability Aligned* – BIAN defines its own particular application of a Service-Oriented Architecture (SOA). The BIAN SOA is based on the definition of discrete business capabilities that present and consume services from other capabilities – BIAN Service Domains. The Service Domains represent elemental building blocks of financial services activity. Their semantic Service Operations make up the interoperability standards that BIAN seeks to define for the industry.

*Capability not Process* – as described there is a subtle but fundamental distinction between modelling business as a collection of processes and modelling business as a collection of service enabled business capabilities. The less common capability representation has been adopted by BIAN to enable the definition of generic elements that can be the basis for defining standards. The business roles and associated service boundaries represented by BIAN Service Domains are stable over time and can be consistently interpreted at different locations.

*Rightsizing Capability Partitions* – to define generic/canonical operational services associated with the BIAN Service Domains, repeatable and traceable design techniques are needed to scope them. The Service Domain's capability partition needs to be both discrete and atomic so that it can be consistently mapped to business activity in any bank. BIAN has defined properties that a well-scoped Service Domain should possess and techniques that can be used to isolate such a capability partition.

## 4.2 The BIAN Service Domain Design and “2 Cycle” Development Approach

The BIAN SOA Design Framework is described in Part 3. It defines the pivotal role of the BIAN Service Domain as the fundamental building block of the BIAN interoperability standards. The semantic definitions of Service Operations at the core of the BIAN standards are associated with Service Domains.

Therefore the concepts and techniques behind the identification and specification of BIAN Service Domains are foundational for BIAN. As already described a BIAN Service Domain is a capability partition that is suited to service enablement. It is an elemental partition meaning it is discrete/non-overlapping with other Service Domains. The collection of all possible Service Domains represents a peer set of capabilities that covers all common banking activity. BIAN Service Domains are generic or canonical, meaning their role can be consistently interpreted across the industry

The current techniques BIAN uses to isolate a Service Domain are described in two categories. The first is a list of required characteristics that can be applied as tests to confirm a candidate Service Domain. The second category defines specific design techniques. Finally in this Section the current BIAN internal working approach is described as used by the BIAN working groups – the “2-Cycle” development approach.

### 4.2.1 Service Domain Operational Characteristics

The following list describes characteristics that are required for all Service Domains, presented in no particular order. At this stage these are defined as expected properties but several are expected to evolve into more rigorous design principles as they are used over time:

- *It has a clearly bounded business purpose* – its business role or purpose is easily defined and associated execution responsibility is ‘assignable’ to some recognizable part of the organization
- *Its business purpose is unique* – its role is discrete from other business activities. Capabilities can be outlined associated with fulfilling this role. (Note that in fulfilling its role, the Service Domain may need to delegate activities to other Service Domains)
- *It is elemental* – it is not an assembly of many finer grained business roles that could be supported by Service Domains
- *It has a ‘focus object record’* – the focus object is a type of business object – it represents a record that captures an instance of the Service Domain fulfilling its business purpose
- *It handles the full life-cycle of its focus object* – the internal capabilities and any delegated tasks handle all possible states of the focus object instance for its full life-cycle (initialization, maintenance, execution, reporting and termination). Note depending on the role and associated focus object the life cycle can be very short-lived or long lasting. An example of long lived focus object is a product design, an example of a short lived focus object is a customer interaction
- *It can handle single or multiple focus object instances* – the role of the Service Domain and its focus object will determine whether it is appropriate for there to be just one or many active instances of the focus object at any one time. An example of a focus object with a single instance is a business plan, an example of a multiple instance focus object is a customer agreement
- *Exclusively service based* – all interactions that a Service Domain has are through offering/consuming Service Operations
- *Loose Coupled* – all service dependencies between Service Domains are loosely coupled, meaning the Service Domains are not ‘locked’ waiting for an expected response but can continue with other activities and process the response when it is provided (and deal sensibly with the situation when an expected response is late or does not occur)
- *Location independent* – if all its service interactions can be supported a Service Domain can be housed in a separate organization
- *Comprehensive business designs* - all possible business activity can be modeled using a selection of Service Domains. (BIAN has attempted to define the full collection of all possible Service Domains – this initial collection will be ratified and extended in use.)

## 4.2.2 Service Domain Design Techniques

There are three discrete Service Domain design techniques used within BIAN:

- 1) Right-sizing the Service Domain's focus object;
- 2) Assigning a single 'functional pattern';
- 3) Confirming its role through business scenarios

### 4.2.2.1 Right sizing the Service Domain's Focus Object

As described in Section 3.1.3.1 "in simple (non-formal) terms" the focus object represents how the Service Domain exerts some type of control over some type of entity (e.g. handles the relationship contract for the entity 'customer'). Rightsizing a Service Domain is also described in Section 3.1.3.2 as defining the narrowest scope where the Service Domain retains "unique business context".

To right size the Service Domain the entity that the Service Domain handles is first isolated. A hierarchal decomposition of the different 'assets' that make up a bank is used to isolate the discrete asset that the Service Domain handles. Assets in this interpretation combine tangible things like buildings and equipment and less tangible things such as knowledge and relationships. The asset decomposition is taken down to the point where the asset is at the finest partition where it retains a meaningful business context.

For example, consider the asset represented by a customer relationship. The asset type 'customer' could be broken down into different categories of customer relationships such as corporate and retail customers at the first level. A decision needs to be made as to how far to take the decomposition into finer categories of customer until the 'lowest common denominator' is found that works for most situations. It might be agreed that it is meaningful to decompose the type 'consumer customer' further to define banking consumers, card consumers and finance consumers.

But to then further decompose the customer type card consumers into credit card consumers and debit card consumers perhaps goes too far. This is where the phrase 'retaining unique business context' applies – it indicates that the distinction between a debit card consumer and a credit card consumer is not sufficient to define two discrete types of customer (or to define two unique business contexts).

BIAN seeks to define the focus object at the level of granularity that matches prevailing behaviors but recognizes that for some Service Domains this determination may need to be refined for specific locations (as outlined in Section 3.2.1.1). A bank may choose to use finer or more coarsely grained entities to suit its own needs, adapting the applicable BIAN Service Domain specifications as outlined in Section 3.2.1.1.



The asset decomposition used is summarised in the diagram:

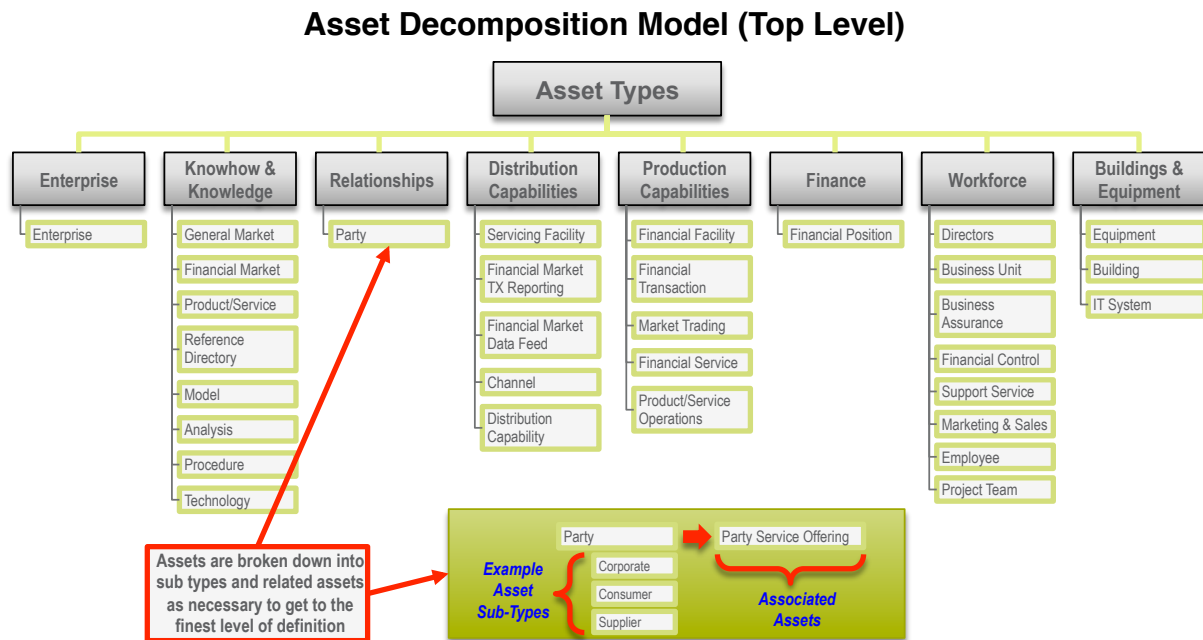


Figure 21 – Asset Decomposition Model

#### 4.2.2.2 Assigning a Single Standard Functional Pattern

A second technique for correctly scoping the Service Domain expands on the property that a Service Domain 'exerts some type of control over some type entity'. Having identified the specific asset handled by the Service Domain, the next step is to determine the single dominant type of control it applies to that asset. Based on an analysis of existing Service Domains BIAN has identified a collection of recurring 'functional patterns' that correspond to the different types of control. These are categorized in the following hierarchy:

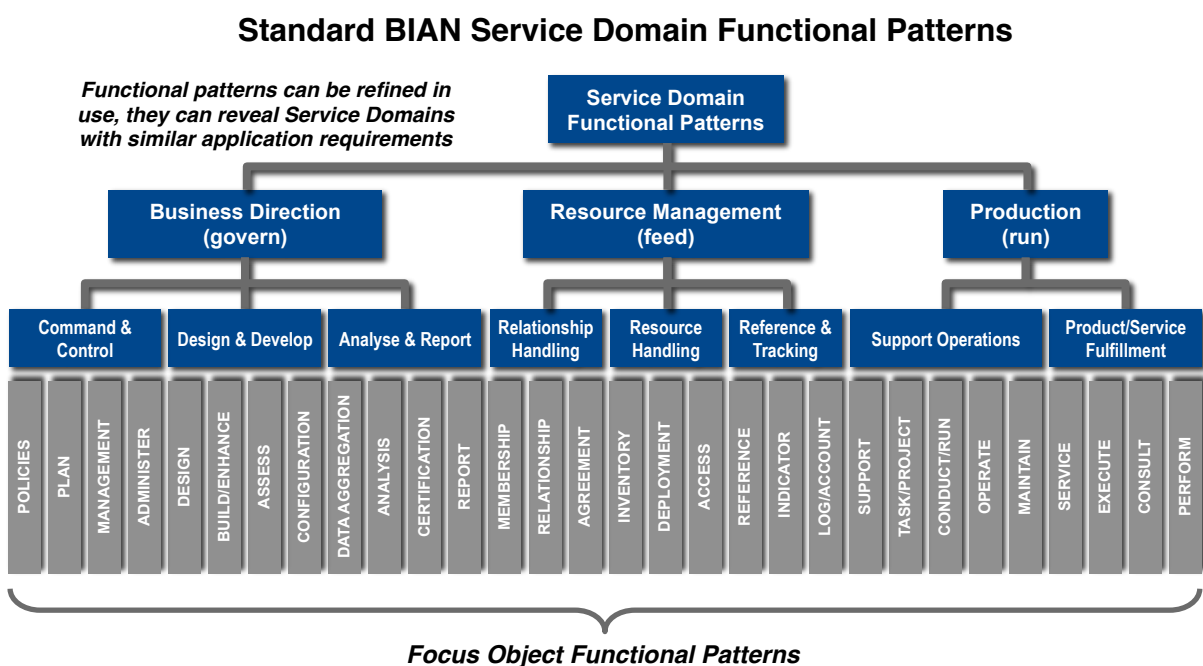


Figure 22 - Functional Pattern Hierarchy

Each of the standard functional patterns is described in a little more detail.

### BIAN Service Domain Functional Patterns Descriptions

Service Domain Functional Patterns	Business Direction (govern)	Command & Control	POLICIES	Define and enforce policies, guidelines and standards
			PLAN	Define and direct the implementation of a plan with supporting budget, organization, goals etc.
			MANAGEMENT	Manage, oversee and troubleshoot an ongoing activity
			ADMINISTER	Handle associated administrative tasks
		Design & Develop	DESIGN	Create and enhance a design, model or specification,
			BUILD/ENHANCE	Build and enhance/extend
			ASSESS	Test, quality assure or assess/evaluate
			CONFIGURATION	Define and maintain the set-up/configuration of a unit or system
		Analyse & Report	DATA AGGREGATION	Consolidate and manage associated data/information
			ANALYSIS	Develop analytical models and derive insights, perspectives from an analysis of data
			CERTIFICATION	Evaluate and certify or rate
			REPORT	Create and provide scheduled and ad hoc (formal) reporting
	Resource Management (feed)	Relationship Handling	MEMBERSHIP	Maintain and administer a membership group/population
			RELATIONSHIP	Maintain a relationship often with supporting goals/plan
			AGREEMENT	Establish and maintain a governing agreement for some form of relationship (can be legal)
			INVENTORY	Maintain and administer an inventory of a collection of items
		Resource Handling	DEPLOYMENT	Deploy, distribute or assign
			ACCESS	Handle access or usage entitlements
			REFERENCE	Capture and maintain reference details for some entity, collectively details can define a directory
			INDICATOR	Maintain status indicators (can be derived/interpreted values)
		Reference & Tracking	LOG/ACCOUNT	Build and maintain a history, log or account of activity (action & financial)
			SUPPORT	Assign specialist internal support resources to address a specific requirement
			TASK/PROJECT	Define and execute a task or project with team, execution plan and deliverables
			CONDUCT/RUN	Provide/run the associated ongoing service/support activity
	Production (run)	Support Operations	OPERATE	Operate a largely machine based capability, optionally to a schedule
			MAINTAIN	Perform maintenance to schedule and repair as necessary
			SERVICE	Fulfill an in-force facility/service delivery commitment – with both scheduled and ad-hoc activity
			EXECUTE	Complete a financial/commercial transaction (includes both short and long life instruments)
		Product/ Service Fulfillment	CONSULT	Provide expertise/advice/support as a service to a customer
			PERFORM	Execute a business function – typically a production support requirement

Figure 23 - Functional Pattern Descriptions

The functional patterns provide a checklist for confirming the Service Domain supports a well-defined and singular operation on its focus object. As with all BIAN techniques the list and definition of allowed functional patterns will evolve with wider use.

#### 4.2.2.3 Confirming the Service Domain Role using Business Scenarios

BIAN makes use of Business Scenarios to clarify and confirm the roles of Service Domains as described in Section 3.1.2. The use of Business Scenarios is also covered in the following section on the “2-Cycle” development approach.

#### 4.2.3 The BIAN “2 Cycle” Development Approach

This Service Domain development approach has two connected design cycles. The business scenario specification cycle supports the less formal interactions with business practitioners. The Service Domain specification cycle is a ‘centrally’ coordinated activity to refine the development of Service Domain definitions including the BIAN Service Domain’s standard Service Operations

Two templates are used in this approach – the Business Scenario template and the Service Domain template (described in more detail in Sections 3.1.2 & 3.1.3 respectively).

The general flow of the approach is as follows:

- Draft Service Domain specifications are used to target candidate Business Scenarios at selected areas.
- The Working Groups (holding interviews and working sessions with business practitioners) refine the scenarios, capturing Service Operation descriptions in the Business Scenario template (Business Scenario Cycle)
- The business scenario interaction descriptions are matched to the associated Service Domain specifications and their Service Operations and input/output parameters proposed. The matched Service Operations are cross-referenced back to the scenarios (Service Domain Cycle)

- The cross reference is used by the Working Groups to confirm the more detailed Service Operation designs proposed, in particular to agree/expand input and output parameter lists (with business practitioner support)

The general flow and templates involved are shown in the Service Domain and their Service Operation specifications are expanded and refined as more 'roundtrip' design iterations are applied using the approach.

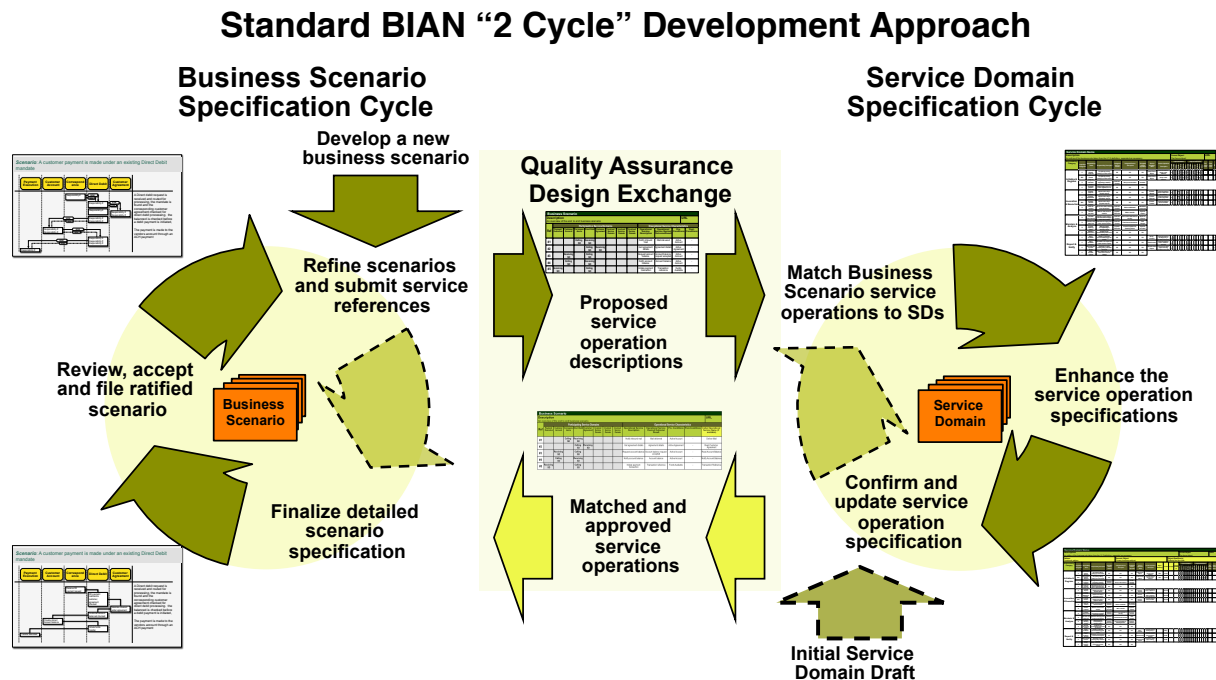


Figure 24 - 2 Cycle Development Approach

### 4.3 The BIAN Metamodel & Business Vocabulary

BIAN has captured its design approach and concepts in a comprehensive Metamodel. Some of the key concepts reflected in the Metamodel are summarized here to provide basic context. BIAN intends that all designs align with the BIAN Metamodel for validity and consistency, and to facilitate tooling support. The BIAN Metamodel extends the ISO 20022 Metamodel; in particular, the BIAN Metamodel's elements supporting the modelling of business objects and service messages heavily leverage ISO 20022.

Highlights from the BIAN Metamodel include:

- It establishes that the Service Domain is an elemental structure – all Service Domains are 'peers': one Service Domain cannot contain finer grained Service Domains.
- Service Domains can be classified by different types/properties and assembled into a reference structure called the BIAN Service Landscape that defines business areas and nested business domains that contain the Service Domains.
- A Service Domain is 'responsible' for the correct execution of its business purpose or role. To do this it has its own internal 'capabilities' and may, through services, 'delegate' activity to other Service Domains as necessary. The BIAN Metamodel formalizes the association between a Service Domain's responsibilities, capabilities and delegation.
- A Service Domain has an associated 'focus object.' It is responsible for all active instances of the focus object for all its possible states (or for its complete 'life cycle'). The focus object is a type of business object.
- A Service Domain has associated services that are organized into service groups
- A Service Operation can be associated with a message – where the message defines the data content of the service and could be mapped to a published message standard.

Basing the BIAN Metamodel on the ISO 20022 Metamodel where there is overlap eases integration of the BIAN semantic service standards with the ISO message standards.

Some fields in the BIAN Service Domain Template have been included to capture specific details needed to support the semantic definitions in the BIAN business vocabulary. These are colored yellow in the template and can be suppressed for certain audiences to avoid confusion.

### BIAN Service Domain Template with Technical Fields

Service Domain Name																												
Description													Focus Object			Reference Object(s)		URL										
An outline of the business role taken from the V1.5 definition, expanded as necessary													The focus object			Reference Objects												
Pattern						Generic Object						Object Qualifier(s)																
Functional pattern						Primary business subject						Reflects the life-cycle																
Responsibility Category	Capability Delegation	Responsibility Name	Responsibility Description	Delegated Service Domain	Delegated Service Description	Delegated Service Operation	Service Operation Name	Service Operation Description	Object Qualifier(s)	Action	Property	Property Qualifier(s)	Input Parameters					Output Parameters					Pre-conditions	Post-conditions	URL			
													Identifier	Status	Control	Address	Age	Identifier	Status	Control	Address	Age						
Initialize & Register	C	Process New Record Request	The primary activity setting up a new record	NA	NA	NA	Create Focus Object	A service used to set up a new instance	New	Create	-	-	ID	Active	Address	Control	Address	Age	Address	Active	Address	Control	Address	Age	Report	-	-	URL
	M/C	Process New Record Request	The primary activity setting up a new record	NA	NA	NA	Create Focus Object	A service used to set up a new instance	New	Create	-	-	ID	Active	Address	Control	Address	Age	Address	Active	Address	Control	Address	Age	Report	-	-	URL
	D	Request customer account	Arrange for the setup of a supporting customer account for the facility	Customer Account	Set up a new account	Create customer account																						
	C	Check for possible duplicates	Check history to see if a similar application has been processed/rejected	NA	NA	NA																						
Invocation & Execution	C	Process Deposit Request	Capture and verify a deposit request	NA	NA	NA	Process Deposit	A service used to make a deposit to the account		Execute	-	-	ID			Amount				ID			Result			-	-	URL
	C	Process Withdrawal Request	Capture and verify a withdrawal request	NA	NA	NA	Process Payment	A service used to make a payment from the account		Execute	-	-	ID			Amount				ID			Result			-	-	URL
	M/C	Process Transfer Request	Capture and verify a payment request	NA	NA	NA	Process Transfer	A service used to make a transfer from the account		Execute	-	-	ID			Amount				ID			Result			-	-	URL
	D	Debit the account	Check and extract amount from the balance	Customer Account	Make a withdrawal	Withdrawal to account																						
	D	Transfer payment	Make an inter-account transfer	Payment Execution	Make a transfer	Payment request																						
Maintain & Analyze	D	Calculate daily interest	Get the account activity and calculate and apply end of day interest	Customer Account	Get account statement for the day	Report on account																						
	D	Calculate and apply fees	Determine and apply fees to the current account	Customer Account	Post fees to the account	Post charges																						
	D	Execute Standing Orders	Make payments corresponding to standing orders	Payment Execution	Process Payments for Standing Orders	Payment execution																						
	C	Maintain account tax position	Update the tax accumulated positions for the account	NA	NA	NA																						
Report & Notify	C	Query Customer Account	Identify a customer account matching given details	NA	NA	NA	Query Account	A service used to identify/query a customer account		Query	-	-	ID	Inactive						ID	Inactive					-	-	URL
	C	Read Customer Account	Assemble and respond to a read account request	NA	NA	NA	Read Account	A service used to read a customer account		Read	-	-	ID	Inactive						ID	Inactive					-	-	URL
	C	Get Customer Account Report	Assemble a report covering many customer accounts	NA	NA	NA	Read Account Report	A service to read a report covering customer accounts		Read	-	-	ID	Inactive						ID	Inactive					-	-	URL
	C	Notify Account Update	Notify subscribers on the event of an account change	NA	NA	NA																						

Figure 25 - The Service Domain Template With Yellow Fields

The technical fields that support the BIAN business vocabulary are:

<b>Functional Pattern</b>	is one of 24 identified functional patterns that BIAN has identified that reflect the type of control exerted by the Service Domain
<b>Generic Object</b>	Is the entity that the Service Domain exerts its control upon
<b>Object Qualifiers</b>	Can provide additional clarification/definition of the Generic Object, both at the overall Service Domain level and specific to a service operation
<b>Action</b>	is one of a finite set of action terms that reflect the nature of the specific Service Operation
<b>Property</b>	is a refinement of the action term, further defining the nature of the action initiated by the Service Operation
<b>Property Qualifier</b>	further qualifies the property for the Service Operation

These terms and examples of their use are provided with the BIAN business vocabulary guideline and technique documentation.

## 4.4 Positioning BIAN alongside Other Standards Organizations

BIAN was established to address a perceived gap in available Banking industry standards. The exchange of information between banks is well-charted territory for which standard services, message content and communication facilities have evolved. The standard business to business, "B2B" solutions have supported significant improvements in operational efficiency and flexibility for financial institutions. However within the banks themselves there are few equivalent service and message

standards. These would define the internal working and exchange of information between operating units and their supporting systems, or “application to application” (A2A). Furthermore though it is accepted that each financial institution must define and execute certain differentiating activities to compete, it is also broadly accepted that much of what goes on inside any bank is common to every bank.

BIAN is the first banking standards body to define semantic service standards with specific focus on internal A2A activity. As noted, most of standards activity has focused on B2B requirements and this has been at the lower ‘implementation’ level. The diagram below shows how BIAN is positioned alongside other standards groups that are active in Financial Services. It is the policy of BIAN to align and contribute to service standards where they exist and not to develop competing standards. BIAN has and will continue to test and confirm that its semantic standards can be mapped to existing standards as appropriate. In particular BIAN seeks to maintain close alignment and integration with the ISO 20022 standards initiative.

## Standard BIAN Positioned Alongside other Standards Bodies

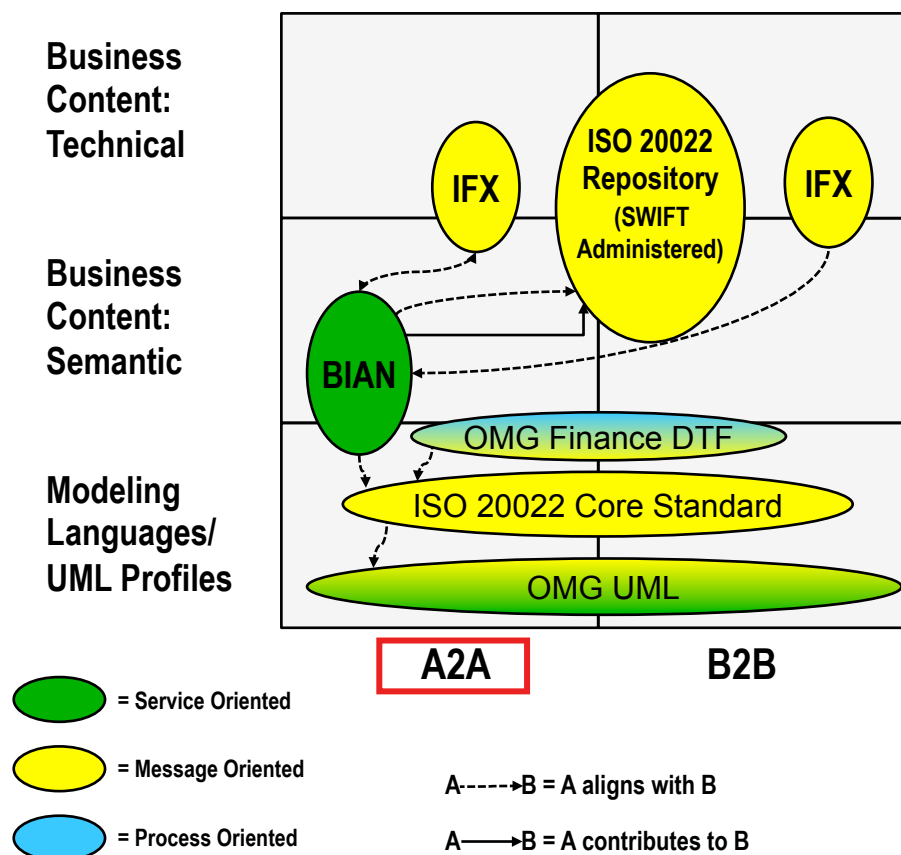


Figure 26 - BIAN in a Matrix with Other Standards Groups

BIAN and the Open Group have issued a joint white paper looking at TOGAF as the overall framework for architecture work and at BIAN as a pool of industry specific architecture deliverables. The white paper analyses how the BIAN deliverables are related to the TOGAF framework in order to accelerate the work of Enterprise Architect working in a Banking environment.

## Appendix 1 - Glossary

Artefact / Artefact	Tangible bi-product produced during the development of the service landscape
Business Area	A broad area as might be associated with a major division of a bank
Business Domain	A smaller coherent set of capabilities within a business area that might correspond to a business unit or specialist business function
Business Scenario	A simple depiction of how selected BIAN Service Domains might work together for some business event
Business Scenario Template	A structured document used by BIAN to define a Business Scenario
Business Vocabulary	A vocabulary in business terms that supports the semantic definition of the Service Domains and the Service Operations
Core Banking	The main transaction processing systems that support banking activity, usually refers to accounting, product processing and CRM applications
Financial Services Industry	Industry which encompasses a broad range of organizations that manage money, including credit unions, banks, credit card companies, insurance companies, consumer finance companies, stock brokerages, investment funds and some government sponsored enterprises. (This definition is from Wikipedia)
Focus Object	A type of 'business object' that reflects the role or business purpose of the Service Domain. In simple (non-formal) terms the focus object represents how the Service Domain exerts some type of control over some type of entity (e.g. handles the relationship contract for a customer)
Functional Pattern	One of a standard list of general actions defined by BIAN that is executed by a Service Domain on the entity that it handles
Functional Pattern Description	A description of one of the standard list of general actions defined by BIAN
Functional Pattern Hierarchy	A classification structure used to group and define the collection of functional patterns
Functional Variation	BIAN defines the general functions and service operations for a Service Domain – it is anticipated that these general terms can be augmented and enhanced to reflect more detailed and location specific functional needs
High Level Semantic Designs	Are the natural language descriptions of business functionality and service operations captured at the 'conceptual' level of detail
Insourcing	Is the practice of offering services to other organization, in the context of this guide, refers specifically to the services associated with one or more Service Domains
Metamodel	Is a formal representation of the design definitions and representations covering all aspects of the BIAN Service Landscape and its supporting artefacts
Non-functional Properties	Are features associated with a design and its supporting systems that address aspects other than the functioning logic, such as performance, availability and security levels.
Object Oriented Analyses and Design	Is a software engineering approach that models a system as a group of interacting objects. Each object represents some entity of interest in the system being modelled, and is characterized by its class, its state (data elements), and its behavior.
Other Standards Group	Refers to other established bodies active in the definition of technology related standards applicable in the Financial Services industry
Outsourcing	Is the process of contracting a business function to someone else, in the context of this guide, referring to the services provided by one or more Service Domains
Point Solution	Refers to a solution with narrowly defined scope that can typically be defined using one or more high level business processes. It contracts a solution that encompasses the whole enterprise or some significant division



Properties	A term referring to specific types of characteristics that apply to elements in the various BIAN design artefacts
Rightsizing	An approach taken to define the optimum scale and scope for some activity or function, which can involve reducing or merging existing elements
Semantic Definition	A natural language definition of a concept or approach/method described to a level of specificity that avoids any implementation specifics
Semantic Service Operations	Semantic descriptions of the interactions between service domains, both offered and called. The service operations define these interactions in simple and general terms. Their implementation can involve far more complex underlying message exchanges
Service Domain	The smallest element of the Service Landscape corresponding to a specific business need or function
Service Domain Mechanics	A schematic representation of the conceptual workings of a Service Domain, used to outline its role or purpose. It does not imply any specific implementation approach for supporting systems
Service Domain Role	A description of the business purpose fulfilled by a service domain, used to encapsulate the main function or requirement satisfied
Service Domain Specification	A reference to the designs of a Service Domain, typically captured in a design template that details the Service Domain's main functionality, the Service Operations it offers and the Service Operations it consumes from other Service Domains
Service Domain Template	A formatted template defined by BIAN to capture the Service Domain design.
Service Landscape	A structured format used to organize the collection of Service Domains primarily for ease of reference. The standard BIAN Service Landscape defines Business Areas and Business Domains within which all Service Domains are uniquely positioned
Service Landscape Working Group	An organized working group within BIAN that is responsible for the definition and content of the BIAN Service Landscape, including the definition and designs of all Service Domains
Service Operation	A term referring to the external access provided by a Service Domain to enable other Service Domains to invoke its capabilities and/or reference business information it governs
Service Partitions	A term referring to the way the range of capabilities that span the scope of a Financial Institution can be divided into non-overlapping and discrete elements
Service Standard	The definition of any aspect of design that is deemed to be commonly interpretable by any industry participant (sometime referred to as canonical)
SOA	Service Oriented Architecture - is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of codes and/or data structures) that can be reused for different purposes.
Working Group	BIAN Members who work collaboratively on different BIAN deliverables and/or specific areas of the Service Landscape and associated groupings of Service Domains

## Appendix 2 – Case Studies

---

### 5.1 SunGard

#### 5.1.1 Why BIAN Makes Strategic Sense for SunGard

BIAN is an independent and unique community, providing a global forum for banks, software providers and system integrators to openly discuss the business and technology requirements of the full banking service landscape. Along with leading global banks such as UBS and Credit Suisse and technology industry heavyweights such as IBM and Microsoft, BIAN's 37 members includes SunGard, the world's largest privately-held software and services company.

Serving approximately 15,000 finance customers in more than 70 countries, SunGard provides one of the broadest portfolios of mission-critical software and technology services for the finance industry. Like BIAN, it believes that technology should be driven by business processes rather than dictating them. Historically, financial institutions have built inflexible systems that, by contrast, have made processes work around them or within their constraints.

For both BIAN and SunGard, the starting point of a solution is not technology itself but the underlying complex business problems it needs to solve. So, with shared objectives in terms of developing an SOA framework, SunGard has naturally looked to BIAN for industry validation of its own ideas and plans. More specifically, it has incorporated BIAN's principles into the new architecture of its core banking solution, taking the opportunity to contribute to, discuss and benefit from BIAN's SOA guidelines.

#### 5.1.2 The SunGard Approach to SOA and BIAN

##### 5.1.2.1 The Benefits of BIAN for SunGard and its Customers

As a founding member, involvement and partnership with BIAN has always made strategic sense for SunGard, which believes that open processes and clear standards are vital for banks and beneficial for the software vendors that serve them, especially in light of the financial crisis. In the view of both SunGard and BIAN, every time a bank acts in isolation and recreates a process that is common to the majority of banks it is wasting its time, probably creating a less efficient customer process than it could, and ignoring the wisdom of others. By the same token, every time a software provider turns these common processes into a proprietary, apparently unique code, the banking industry is less well off. Through BIAN, banks and software vendors can instead learn, discuss and disseminate best-process practices.

While BIAN's ever-growing membership of leading banks and technology providers has brought even greater credibility to its architectural approach, the approach itself has already revealed potential benefits for SunGard's customers. As a result, BIAN's SOA framework and Service Landscape was to prove vital in shaping and validating SunGard's new core banking solution, which it launched as part of its Ambit Banking solution suite in 2008.

Applicable to a broad range of banking services, BIAN's Service Landscape helps SunGard articulate its vision to its customers with a new clarity, giving them a blueprint to help plan what their banking system could look like. What is more, it can help banks to visualize their service provisioning and identify areas that might need improvement. In this sense a bank can plan for a phased approach to replace only the gaps in service it needs to fill, and at the same time minimize integration costs, while BIAN-compliant terminology and standards further smooth the integration process.

##### 5.1.2.2 SODA: SunGard's Interpretation of BIAN Guidelines

SunGard's methodology and approach to design and development, i.e. the way it identifies or "discovers" new service solutions, is very much aligned to BIAN's recommendations, so the starting

point for product development is always to take a complex business problem and break it down into elemental building blocks, to the point that it becomes, in the language of BIAN, a Service Domain.

An example of a Service Domain could be Collateral Management, which is applicable to multiple business activities in a bank, from loans to investments. A traditional software design environment might develop separate collateral management processes for each business activity, whereas BIAN and SunGard view collateral management as one “problem” that can be solved by the same Service Domain across all applicable business purposes.

SunGard named its SOA methodology Service-Oriented Discovery and Analysis (SODA). As a high-level approach to finding the right model for a complex problem, and following a continuous lifecycle of modeling, development and monitoring, SODA consists of:

- a set of templates based on SunGard’s five-layer, industry-standard Ambit Reference Architecture and BIAN’s methodology and standards;
- a service landscape;
- a unified set of semantic definitions: not only a common, natural, widely understood language and terminology, but also consistent, universally recognized standards, such as for messages, to maximize the interoperability of solutions.

The table below articulates SunGard’s understanding of the different elements within the service landscape.

Subject	Characteristics
Business Area	<ul style="list-style-type: none"> <li>• Grammar for organizing business domains that constitutes an institution's main business operation</li> <li>• Arbitrary, but necessary for conversation</li> </ul>
Business Domain	<ul style="list-style-type: none"> <li>• Grammar for organizing service domains that constitutes an institution's sub business operation</li> <li>• Arbitrary, but necessary for conversation</li> <li>• Meaning is context driven</li> <li>• Business separable organization unit</li> <li>• Multiple responsibilities, one business purpose</li> </ul>
Service Domain	<ul style="list-style-type: none"> <li>• Grammar for organizing services that constitutes an institution's business function</li> <li>• Unique responsibility, collection of capabilities</li> <li>• One focus object</li> <li>• Handles ever relevant occurrence – “complete”</li> <li>• End-to-end lifecycle</li> </ul>
Service	<ul style="list-style-type: none"> <li>• Single activity in a business function</li> <li>• Distinct purpose</li> <li>• Fails the sub-domain test</li> </ul>
Service Operations and Components	<ul style="list-style-type: none"> <li>• Utility / tool</li> <li>• No independent business value</li> <li>• Inherently re-useable</li> <li>• Loosely coupled to business actions</li> </ul>

Figure 27 - SunGard's Adoption of SOA and BIAN's Service Landscape

Above all, SODA demonstrates SunGard's thorough understanding of BIAN standards and ability to map them from business processes back to technology. Through SODA, and continuing its own well-established design and development practices, SunGard takes both a “top down” and “bottom up” approach to identifying and right-sizing services for its service landscape. The “bottom up” approach is used to validate the identified services against the functions and features of current software assets, and to ensure that the new landscape can accommodate the existing users of SunGard's core banking solution. Having assembled the appropriate services and its contents, it can then refer to BIAN's recommended Service Landscape for validation on completeness and relevance.

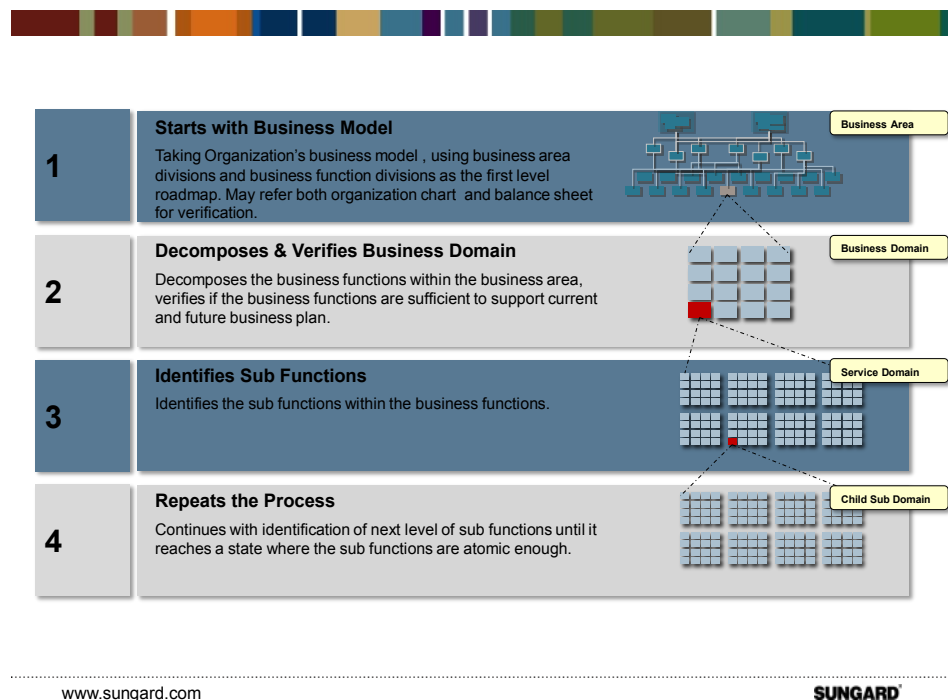


Figure 28 - SunGard's Top Down Approach to Service Identification

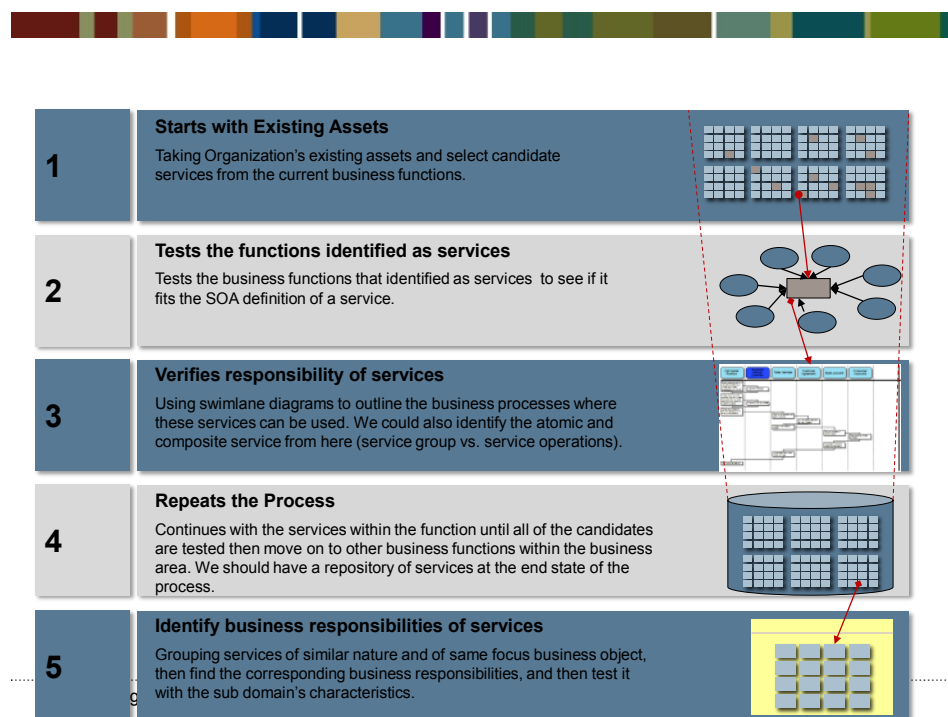


Figure 29 - SunGard's Bottom Up Approach to Service Identification

## 5.1.3 SunGard's Ambit Core Banking

### 5.1.3.1 The History of Core Banking

Over the last 50 years, core banking systems have evolved as a direct consequence of their role and the demands of the banking landscape in general. Once merely a transactional and account-centered

back-office system, core banking architecture has responded to progressive changes in bank strategy. Led by the increasing demands of customers along with greater competition, itself the result of widespread industry deregulation, the scope of core banking has therefore adapted to the changing needs of the time: to be, in turn, product centered, channel centered and most recently customer centered.

The global financial crisis and the ensuing challenges faced by banks has meant that core banking must evolve once more. This time it needs to respond to more rigorous risk management practices, increased regulation and a renewed emphasis on the health and stability of banks' single most important asset: their balance sheet.

In the years immediately following the crisis, far too many banks have found their systems are either not up to the task or simply inadequate when it came to providing executive management, boards and regulators with the transparency necessary to manage the bank in crisis mode. As a result, banks and regulators have identified the need to streamline risk architectures, implement a common data model and ensure a seamless flow of risk management policy and practice across the entire process: from the front office to the middle and back offices.

For mid and small-tier banks, the core banking system is the main, sometimes only, transactional and account processing engine, and therefore the core banking system has become a primary provider of data into risk management systems. Integration of core banking and risk management systems, however, does not always enable greater transparency of the bank's true risk profile. Integration architectures with multiple layers of extraction, transformation and aggregation remove the integrity of data, leading to inaccurate, confusing and often incomplete information, making it difficult for the bank's executive management team and board to steer the organization through the dynamics of the market.

The core banking system is also the primary system used by staff across the enterprise to conduct front-middle and back-office servicing of customer financial transactions and accounts, and is therefore home to many of the daily processes which need to adapt to provide a new level of responsiveness and agility.

As executive management and risk practitioners develop response strategies during times of crisis or indeed in times of ongoing market volatility, the ability to operationalize these strategies through the organization and its processing systems has become a pre-requisite in order to ensure the ongoing stability and safety of the bank. To this end, core banking systems must now evolve to support greater assimilation with balance sheet management tools and risk systems in a two-way flow of data and integration of processes.

### **5.1.3.2 SunGard's Ambit Core Banking solution**

To rise to the above market challenges, and improve process integration, SunGard decided to renew its core banking solution. The result was SunGard's Ambit Core Banking solution: a universal banking system that provides comprehensive instrument and product coverage for retail deposits and lending, commercial banking, trade finance and treasury management. Ambit Core Banking enables banks to more effectively manage increasing margin pressure and growth by helping to optimize the balance sheet profile, develop a strategic balance sheet management framework and operationalize analysis and learning into executable strategies.

Applying the service identification principles of SODA, and reflecting BIAN's SOA approach, SunGard built Ambit Core Banking with a clear architectural vision in mind. First, operating natively in SOA environments, the solution allows for rapid and above all cost-effective functional enhancement. Following the Service Landscape methodology, it can also be sold in any pattern of components and service combinations, allowing for easy interoperability with other component, service and application sets.

Overall, Ambit Core Banking guarantees a stable platform for the ongoing commercial life and development of a bank's core system. In so doing, it also provides a set of agile retail banking services that allow banks to flexibly organize their service operations according to their intended business landscape, as well as meeting ever-changing retail banking business needs.



### 5.1.4 Working with Erste Bank

In 2012, SunGard began a new chapter of partnership with long-standing customer, Erste Bank Group, giving SunGard a real-world opportunity to apply the principles of SODA to requirement gathering and solution development, and test its strategic adoption of BIAN principles. One of central and eastern Europe's leading financial services providers, Erste announced its own membership of BIAN in November 2012.

SunGard's previous core banking system had proved a reliable solution for four Erste bank networks in the Czech Republic, Hungary, Slovakia and Ukraine, but SunGard saw the potential to enhance it significantly by implementing its new SOA-based version, Ambit Core Banking. Furthermore, rather than carrying out a wholesale system replacement, and thereby increasing the cost, risk and timescales of the project, SunGard would take an "evolution not revolution" approach. This step-by-step transition, clearly in line with BIAN guidelines, would help achieve shorter time to market for new functionality, with minimal disruption to business, better control over costs and greater flexibility.

The first phase of the upgrade will take place at Erste Bank in Hungary, followed by other subsidiaries who will upgrade in 2014/15.

Christian Gosch, CIO of Erste Bank, said: "We can adjust implementation plans according to individual business needs, which makes the evolutionary upgrade easier. Behind these priorities is one big plan. Our aim is not to unify all core banking systems, but to unify them in clusters. This is not a core banking replacement project, but instead we are having an evolutionary upgrade, which will be minimally invasive for our different businesses."

This production of tangible solutions by SunGard with Erste marks a real milestone for BIAN and its members. In particular it underlines how the BIAN Service Landscape can provide a blueprint for banks to develop architecture roadmaps, which in turn support the flexible integration of existing applications with new solutions.

### 5.1.5 Conclusion and Next Steps

As an active contributor to and consumer of BIAN's SOA assets, SunGard has embraced BIAN's guidelines and demonstrated how its Service Landscape can help address its clients' banking needs and clearly show them a new, evolutionary way forward for the underlying technology that supports their business operations.

Over this period, however, BIAN's approach, assets and vocabulary have themselves evolved. For example, a UML-based modeling tool is now being used to describe the Metamodel, Service Domains and Business Scenarios, taking the place of basic Word or PowerPoint documents, as a formal representation of BIAN's designs and design concepts, to maintain their integrity and improve tooling support.

Having continually invested and improved its own SOA framework in parallel with BIAN, SunGard is continuing to migrate to its new core banking architecture, maintaining its dialog with BIAN, feeding back and discussing its experience of the service landscape model and the impact this has had for its customers.

Beyond BIAN and the service landscape environment is SunGard's work to optimize its user interfaces, making them easier to use and more visually appealing and intuitive for users. As well as demonstrating SunGard's commitment to meeting the needs of the end user, its clients, and ultimately their own customers, this final layer of development sits well with BIAN's overarching aims to provide technology of the highest quality to the financial services industry.

## 5.2 Scotiabank

Scotiabank's usage of BIAN Service Landscape as a functional classification scheme for SOA Services

Scotiabank has set up a repository for storing metadata of SOA services and the BIAN Service Landscape has been configured to be the main functional classification scheme in the repository (refer to Figure 30 below). The benefit is that an open and comprehensive standard is leveraged so that services can be classified consistently. Without the BIAN Service Landscape, the bank would have to resort to either creating our own proprietary classification scheme or using a vendor provided scheme. The natural consequence of classifying services consistently is to enable the ability to analyze gaps and overlaps as more and more services are developed.

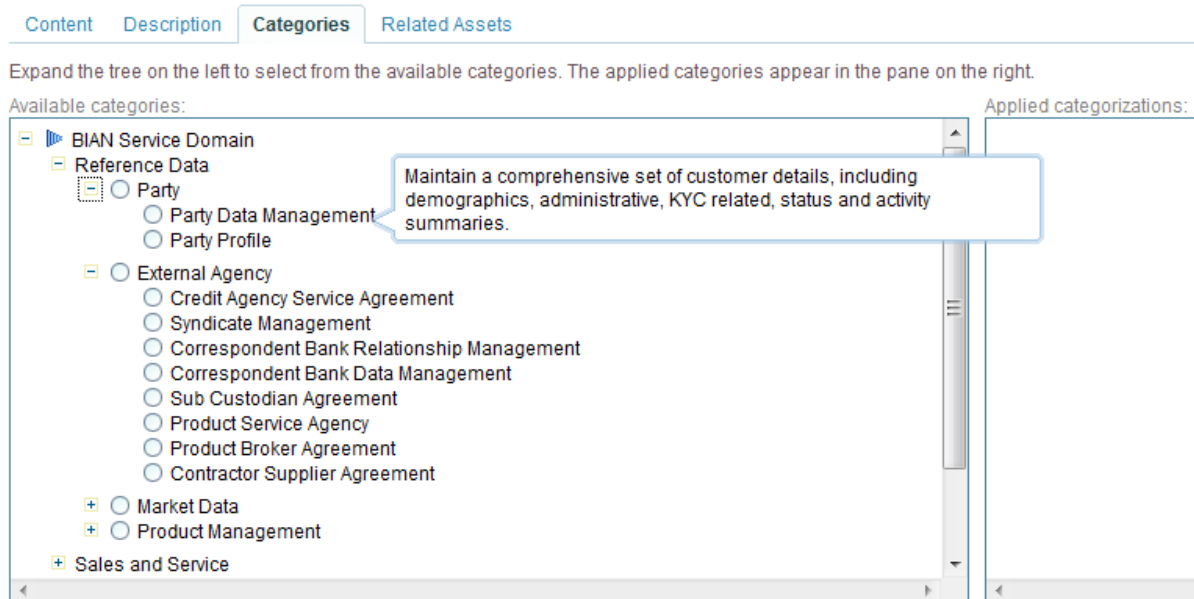


Figure 30 – Scotiabank's Repository Based on the BIAN Service Landscape